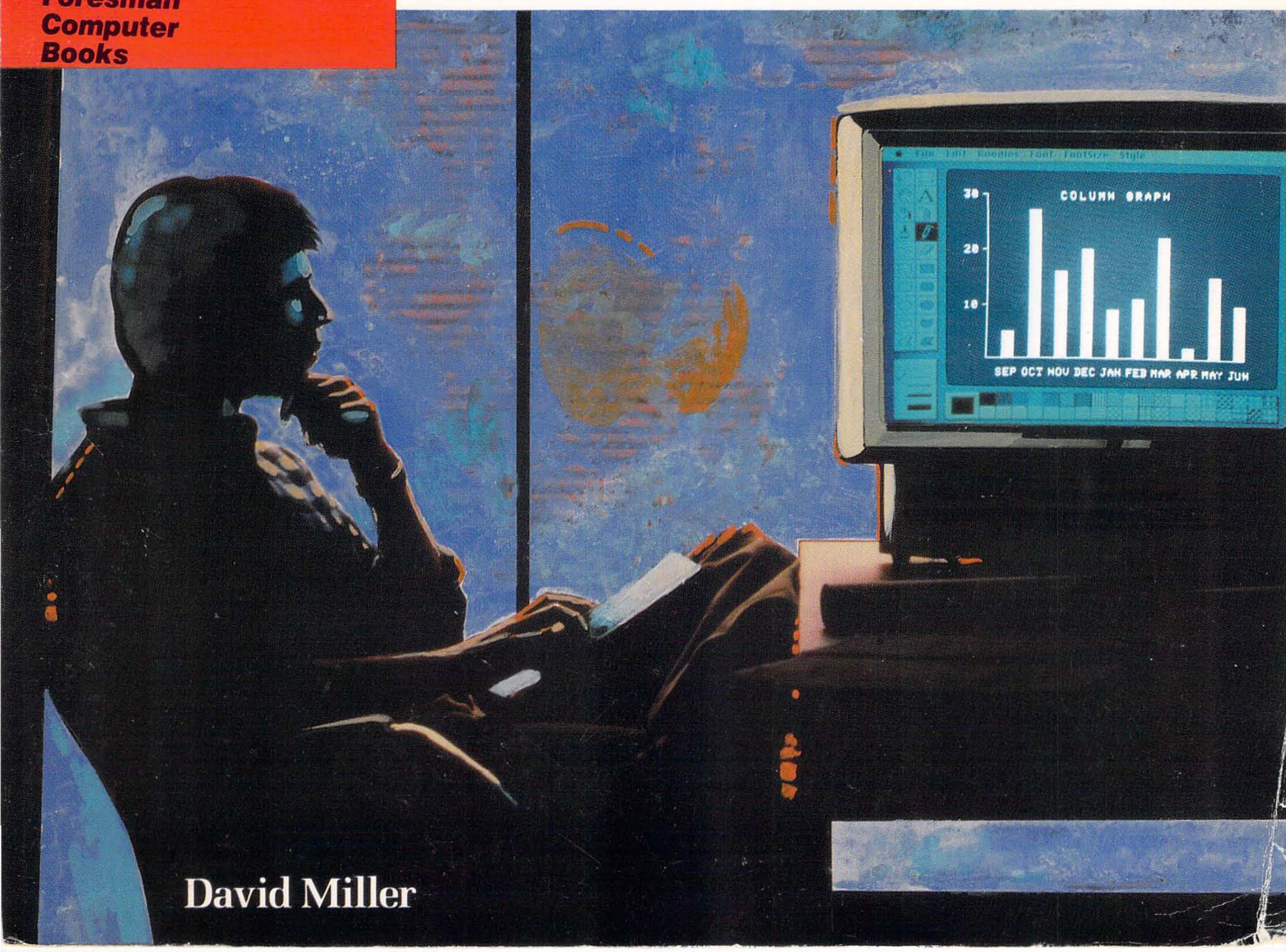
Mastering Applesoft Graphics

Scott, Foresman Computer Books



David Miller

BOOKWARE

The programs in this book are available on diskette(s). Additional file programs and documentation are also included. To order the diskette(s) and documentation, please send $25 ($27.50 outside North America) to:

David Miller
1750 Sulphur Springs Rd.
Corvallis, OR 97330.

Please be certain to specify "Graphics Disk" and the format you wish – 5 1/4″ or 3 1/2″.

In addition to the diskette(s) that contain all the programs listed in this book, we have other application programs. The following systems may be purchased (please add $2.50 for mailing if you are outside North America):

Grading System (an extensive series of programs used to keep track of Student grades) – $50. (Permission is granted to make as many copies *per school* as necessary.)

Substitute Teacher's Report System (used by substitute teachers to leave a report for the teacher they replace; reports are also stored on diskette for future reference) – $25. (Permission is granted to make as many copies *per school* as necessary.)

Mailing List System (a fairly sophisticated and complete random access database system used to store and retrieve names, addresses, and phone numbers; category or code classification is included) – $25.

Stock Market System (used to track price and volume history of stocks) – $50.

We are also prepared to customize a system for your particular needs. Please write for more information about customized systems. Thank you. I hope you find the book enjoyable and useful.

Mastering Applesoft Graphics

David Miller

Scott, Foresman and Company
Glenview, Illinois London

Apple, Applesoft, Macintosh, and IIGS are registered trademarks of Apple Computer, Inc.

1 2 3 4 5 6 RRC 93 92 91 90 89 88

Library of Congress Cataloging-in-Publication Data

Miller, David
Mastering AppleSoft Graphics.

Includes index.
1. Apple II (Computer)—Programming. 2. Computer graphics. I. Title.
QA76.8.A662M55 1989 006.6′765 88-30814

ISBN 0-673-38148-X

Author Disclaimer
After much consideration, I decided that color illustrations would not significantly add to the reader's learning experience yet would significantly add to the cost of the book. I hope that this decision does not offend or deter readers from using the book to create graphics with Applesoft Basic.

Notice of Liability
The information in this book is distributed on an "As Is" basis, without warranty. Neither the author nor Scott, Foresman and Company shall have any liability to customer or any other person or entity with respect to any liability, loss, or damage caused or alleged to be caused directly or indirectly by the programs contained herein. This includes, but is not limited to, interruption of service, loss of data, loss of business or anticipatory profits, or consequential damages from the use of the programs.

Scott, Foresman professional books are available for bulk sales at quantity discounts. For information, please contact Marketing Manager, Professional Books Group, Scott, Foresman and Company, 1900 East Lake Avenue, Glenview, IL 60025.

And may the Lord of peace Himself give
you peace all the time and in every way.
II Thess. 3:16

In loving memory of my father
Harold Miller
who took all the
photographs for this
book. We miss him but
know that he now has
"peace all the time
and in every way"!

Contents

CHAPTER 1

Introduction to Applesoft Graphics

The only feature that has remained the same on the Apple *II*, *II*+, *II*e, *II*c, and now the Apple IIGS is Applesoft BASIC. *Mastering Applesoft Graphics* is designed around that simple fact. Owners of any of these machines can use this book to learn to create graphics on their computer.

Apple has continually expanded the capabilities of the Apple *II* computer since its introduction in 1977. Applesoft BASIC, on the other hand, appeared in 1978 and has remained essentially unchanged. Applesoft programs written in 1978 work properly today on very different Apple *II* computers. Computer owners who upgrade their systems do not have to throw out their software and purchase new programs, and books like this one can present information that is applicable to all Apple *II* machines.

Applesoft routines included in this book that access new capabilities in graphics and sound should prove especially useful to Apple IIGS owners, since Apple has not provided direct access to these new capabilities. Instead, Apple IIGS owners are expected to learn either C or Assembly language in order to use the additional graphic and sound features. Both of these languages are mainly for professional programmers. Using the routines given, you will be able to take advantage of the new features without learning "C" or Assembly language.

I have taken a unique approach in this book, defining computer graphics through the display modes that are available on the various machines. This approach illustrates contrasts among the display modes and among the capabilities of the different computers. Anyone interested in programming graphics must make choices in order to display the work. A wrong choice will produce undesired results and waste many hours. The method used in this book will help you avoid some of these wrong choices.

Apple's display is like a television. If you want to see a different program on TV, you select a different channel by either turning the dial or pressing a button. Either way, you will switch from one display screen to another. The Apple *II* line of computers has four basic display screens, with a number of variations possible for each screen. Enhanced Apple *II*e computers, Apple *II*c computers, and Apple IIGS computers have additional display modes.

Each of the four basic screens has different functions in much the same way that some TV channels are dedicated to one type of programming – sports, religion, news, movies, etc. The first screen (or channel) is capable of showing full-screen, 40-column text material (like this page, but with only 40 characters per line) and low-resolution (large-block) graphics. The second screen is similar, in theory, to the first screen; it also can show text and low-resolution graphics. In reality, it is rarely used for either. The third screen is for high-resolution or small-dot graphics. Four lines of normal text can also be shown on this screen. The last of the four screens is similar to the third screen in that it is dedicated to high-resolution graphics, but it usually lacks the option of showing four lines of normal text. The four basic screens are:

1. 40-Column Text/Low-Resolution 1
2. 40-Column Text/Low-Resolution 2
3. High-Resolution 1
4. High-Resolution 2

Enhanced Apple *II*e and Apple *II*c computers have three additional display modes: an 80-column text screen (80 characters per line), and double high- and double low-resolution screens. The three additional screens are

5. 80-Column Text
6. Double Low-Resolution
7. Double High-Resolution

Apple *II*GS computers include the four main screens, the three additional screens and add two super high-resolution screens:

8. Super High-Resolution 1
9. Super High-Resolution 2

As you can see, Apple IIGS owners have the most options (9), and Apple *II* and Apple *II*+ owners have the least (4). Regardless of the total number of choices, all Apple *II* owners have access to the four main screens. Applesoft BASIC, included in all versions of Apple *II* computers, makes it possible to write programs that will run on any Apple *II* computer. The next chapter begins the discussion of Applesoft Graphics by briefly examining the text/low-resolution screens.

CHAPTER 2

The Text/Low-Resolution Screens

Text/Low-Resolution 1 or Page 1 is the most common screen display, since it is dedicated to showing normal text material. This page is also capable of displaying low-resolution graphics instead of text or low-resolution graphics with four lines of normal text at the bottom of the screen. This first display screen is often referred to as the **Text/Low-Resolution Screen** or **Page 1.**

The second screen has the same rules and capabilities as the first screen. In theory, it also can show all text, all low-resolution graphics or a combination of low-resolution graphics and four lines of text. In reality, this screen is rarely used for either text or low-resolution graphics, because it shares its location in the computer's memory with the instructions contained in Applesoft BASIC programs. In order to use this screen for anything other than Applesoft programs, you must give the computer very specific instructions as to where you want the text/low-resolution information or the Applesoft programs to be stored in memory. Specific directions are necessary because you are changing the place in memory where Apple normally stores these different functions. This second screen is called **Text/Low-Resolution Screen 2, Page 2,** or **Program Memory.** Double low-resolution (or medium-resolution, as it is now often called) refers to the use of either of the first two display screens together with additional memory to increase the display capability of the low-resolution screen by a factor of two. In other words, twice as much information can be placed on the medium-resolution screen. This display is also not easy to use, since Applesoft BASIC does not directly support double low-resolution graphics. (With all the other options available, double low-resolution graphics are more trouble than they are worth.)

SCREEN-DISPLAY TUTORIAL

The tutorial begins by showing the steps necessary to see some of the screen displays. We will skip Text Page 2 for now, since the use of that page for anything other than Applesoft BASIC program storage involves commands that will not be covered until later in this book. Double low-resolution graphics involves the same commands and therefore will not be included in this tutorial either.

Start up the computer and disk drive. (If you are using a computer with 80-column capability, place the computer in the 40-column mode, and depress the CAPS LOCK key so that what you type will be uppercase.) Use a diskette that allows you immediate access to the Applesoft BASIC system prompt and cursor:

```
] []
```

If you are not sure how to get the system prompt and the cursor on the screen, check the manuals that came with your computer. There are too many different Apple *II* computers and operating systems for me to give specific instructions for them all here. Therefore, in this book, I will assume that you can start up your system and get the system prompt on the screen.

The problem you are most likely to encounter in following the instructions in this book is typing errors. Because computer keyboards contain additional keys or keys in different locations from other keyboards, no matter how experienced a typist you are you are likely to make mistakes. Since you are not typing on the screen but, instead, are typing into the computer's memory, because of typing or editing errors, what appears on the screen may not actually be what is in the computer's memory. Therefore, you should try retyping several times before deciding that something is wrong with the machine or the instructions. The instructions given here have been successfully used with a number of beginning Apple users. If you follow them carefully and accurately, correcting any typing errors or retyping and trying again when something does not appear to work, you *will* see the intended displays. Do not become discouraged if you find yourself retyping many times. As I said, **everyone** experiences some trouble finding and pressing the right keys.

When it is first turned on, the computer has a comparatively limited understanding of words or phrases typed into it from the keyboard. For example, type the word "hello" and press the RETURN key:

```
HELLO {RETURN}
```

The word RETURN with brackets around it indicates that you are to press the key marked RETURN. Do not type the letters R-E-T-U-R-N. (Some Apple keyboards have the return key marked with capital letters, while others have that key marked with lowercase letters. In this book, I will always refer to keys to be pressed with all capital letters.)

The computer should respond with

```
?SYNTAX ERROR
```

and a sound or beep of some sort. This message indicates that the computer did no understand this English-language greeting. Instead, type:

```
PRINT "HELLO" {RETURN}
```

This time the computer responds with

```
HELLO
```

directly below the instruction. The computer does this because you have used one of the words in its BASIC language vocabulary, PRINT, and have used it in an acceptable way (i.e., with quotation marks around the message you want displayed). This limited vocabulary with exact syntax rules is another cause of initial user error. Once again, do not become discouraged if it appears that you are following the instructions exactly but are not achieving the desired results. Sometimes the difference between success and frustration is simply the difference between a comma (,) and a semicolon (;). I have spent hours reviewing the logic of computer instructions, looking for the reason a program does not work as it should, only to discover a line that contained the letter O when it should have contained a zero (0). Though I understood the intended instruction, the computer could not understand it and continued to indicate an error. Since the computer does not have any assumed knowledge there is a big difference to the computer between 0 and O.

Now, if you are not completely discouraged, let's try typing some instructions. Remember that after you finish typing, you must press the key marked RETURN. Type

```
PRINT "HELLO, I AM AN APPLE COMPUTER." {RETURN}
```

(Press the RETURN key after you have typed the last quotation mark.) You should see the following on the screen:

```
]PRINT "HELLO, I AM AN APPLE COMPUTER."
HELLO, I AM AN APPLE COMPUTER.
```

Unless you have cleared the screen there will be some other words also, but we are concerned with the words given above. You have just entered and displayed information on the Text/Low-Resolution Screen or Text Page 1. This first screen can also display low-resolution graphics or block graphics. To see this aspect of the first display page or screen, type

```
GR {RETURN}
```

The screen should clear and the cursor drop to the bottom of the screen. If it doesn't, type the GR instruction again. Now, it is virtually impossible to use the word PRINT to display anything in the top 20 lines of the screen. (I try to avoid making absolute statements, because I know that some super programmers delight in achieving what is supposedly impossible.) The point here is that the word PRINT is not intended to be used in the graphics portion of either Page 1 or Page 2 while the low-resolution graphics mode is in effect. Because the GR instruction provides for four lines of text at the bottom of the screen below the low-resolution portion, you can still use instructions intended for use with text material (such as PRINT) in those four lines. But to see something within the first 20 lines, you must use other computer instructions. Type

```
COLOR = 6 {RETURN}
```

Nothing happens! The screen does not change colors. But you have instructed the computer which of its 16 low-resolution colors you want to use (6 designates the color blue). Continue by typing

```
PLOT 20,20 {RETURN}
```

This time something happens. A blue block shows up approximately in the middle of the screen. (On some display devices it may appear as nearly a square, while on other display devices it will appear in its true rectangular shape.) Again, if you do not get this block to appear on your display device (television or monitor), check your typing and try again until you succeed. Different display devices vary widely in their color-display capabilities. In this book, all references to specific colors will correspond to the color code given by Apple Computer Inc. in its manuals (see Table 5.1).

You have now displayed information on Page 1 while it was in its text mode, and displayed a block graphic while it was in its low-resolution mode. Now that it is in its graphic mode, the text created earlier ("Hello, I am an Apple computer") is no longer visible. This first screen or page will not retain what is in text when you switch to graphics or retain what is in graphics when you switch back to text. To see that this is true, type:

```
TEXT {RETURN}
```

Suddenly, with the exception of the tenth line, all the top lines contain inverse @ signs. The one exception on line ten is a point that contains an inverse F. You may have to look very closely to see this, but one of those positions should contain the character F. Notice also that the position of the F coincides exactly with the position of your blue block. This should provide further evidence that, on this first screen, you can see either text or graphics but not both at the same time in the same screen location. The area of the computer's memory in which the page is stored contains rules that allow the computer to show either text or low-resolution graphics (or a combination that divides the screen into graphics and four lines of text) but does not usually allow both to be displayed simultaneously.

To get rid of the inverse @ signs, use another instruction designed for the text screens. Type:

```
HOME {RETURN}
```

Now the screen should clear, and the cursor should be in the upper left-hand corner. If this is the case on your display device, you are ready to try answering the questions at the end of this chapter. If your screen is not clear, with the cursor in the upper left-hand corner, try typing the HOME instruction over again.

In the next chapter, we will look at how we can access information on the two high-resolution screens. If necessary, before moving on to Chapter 3 go back over the information in this chapter until you are able to answer the questions that follow without looking back for any answers.

Table 1 in the Appendix summarizes Text/Low-Resolution Page 1 and Page 2 graphics.

REVIEW QUESTIONS

1. What are the two main display modes of Page 1?
2. True or False? Page 2 is normally used to store Applesoft BASIC programs.
3. True or False? Apple computers understand the English-language greeting "Hello."
4. What is one of the instructions used to display information on the text screen?
5. Which instruction informs the computer that it should switch to displaying low-resolution graphics?
6. Which instruction is used to set the color used in low-resolution graphics?
7. Which instruction allows the user to display a single low-resolution point on the screen?
8. Which instruction returns the computer to text display?

CHAPTER 3

The High-Resolution Screens

The third main Apple screen can show high-resolution (small dot) graphics or a combination of high-resolution graphics and four lines of text. Some confusion can occur over the name of this third display screen since it also is sometimes referred to as Page 1. Usually, however, it is called the High-RESOLUTION screen (or simply Hi-Res) Page 1. This is analogous to a television the first channel of which is called Channel 1, the second channel of which is known as Channel 2, and the third channel of which is referred to as Channel 1-H. In both the television and the Apple computer, there are three different displays, two of which share a portion of their names but are very different types of screens. To avoid confusion, in this book I will refer to the Text/Low-Resolution screen as Page 1 or Text Page 1 and the High-Resolution screen as High-Resolution 1 or High-Res Page 1.

Completing the Apple's normal display capability, the fourth screen operates in much the same way as the third screen. This display can also show high-resolution graphics and, in theory, can show a combination of hi-resolution graphics and four lines of text. Usually, though, this display is only used to show a full screen of high-resolution graphics, because its display of text must come from Text Page 2, which shares its place in the computer's memory with Applesoft BASIC programs. This fourth display screen is called High-Resolution Screen 2 or Hi-Res Page 2.

Double high-resolution is not directly available from Applesoft BASIC. Like double low-resolution, double high-resolution increases the amount of information that can be shown on either of the main high-resolution pages by a factor of two. The method used to accomplish both double low- and double high-resolution involves more memory than the normal 64K available to 8-bit machines. The need for additional memory and the fact that Applesoft BASIC does not contain reserved words designed for either type of display are the main reasons double low- and high-resolution graphics are difficult to use.

SCREEN-DISPLAY TUTORIAL

To begin the tutorial part of this chapter, start up the computer and disk drive. (For those using a computer with 80- column capability, place the computer in the 40-column mode and depress the CAPS LOCK key. Use a diskette that allows you access to the Applesoft BASIC system prompt and cursor.

For those who may have skipped the first two chapters, I will repeat the statement made in Chapter 2 regarding the procedures for starting up the computer. If you are not sure how to get the system prompt and cursor on the screen, check the manuals that came with your computer. Because there are a number of different Apple *II* computers and operating systems, the specific instructions for starting the computer and disk drive are too cumbersome to include here. Therefore, I will assume that you are able to start up your system and get the system prompt.

When you have the cursor (blinking box) on the screen, press the RETURN key until it is at the bottom of the screen. (This step is very important—otherwise you will not be able to see what you are typing.) Then type

```
HGR {RETURN}
```

(Remember, {RETURN} indicates that you are to press the key labeled return, not that you are to type the letters: R-E-T-U-R-N.)

The screen will clear and the cursor will be blinking in the lower left corner. In fact, the display should look almost the same as it did when you used the GR instruction in Chapter 2. The only real difference will be that different letters may be showing within the four-line text "window" at the bottom of the screen. When you issued the GR instruction, you may have seen the letters GR still showing just below the edge of the low-resolution portion of the screen. This time the letters HGR may still be visible.

There is a big difference between the two displays, though. If you have typed correctly, the instructions we used to get the blue block will be useless on this screen. Just to make sure, type

```
COLOR = 3 {RETURN}
PLOT 20,20 {RETURN}
```

Nothing happens. No error message occurs because COLOR and PLOT are valid Applesoft instructions, but they are not instructions that are associated with high-resolution graphics. Even though the screen may look the same, we are accessing a different part of the computer's memory and will get a very different type of display when we do use the proper instructions. Although they produce remarkably different results, the instructions used for high resolution are quite similar to those for low-resolution. The only difference between the two sets of instructions is the letter H. This time type

```
HCOLOR = 3 {RETURN}
HPLOT 20,20 {RETURN}
```

Look carefully! Now, instead of a large, colored block in the center of the screen, there is a very small dot in the upper left portion of the screen. This dot is the size of the smallest unit available on the high-resolution screens. The colored block demonstrated in Chapter 2 was the smallest unit available on the low-resolution screens. (Again, I am speaking of what is normal rather than what is possible.)

High-Resolution Screen 2

You have just entered and displayed information on Apple's third screen, High-Resolution Screen 1, also known as the primary high-resolution screen. Next we will enter and view information on the last of Apple's four main screens—High Resolution Screen 2.

This next instruction may cause you some alarm at first, but don't worry: you cannot hurt the computer by any instruction you give it (there is no self-destruct instruction!). The computer is okay even if you cannot see anything on the screen. (**Do not** attempt to follow the next set of instructions if you have been using the 80-column mode. The Applesoft HGR2 command may not work properly with the 80-column mode active.) Type

```
HGR2 {RETURN}
```

Did the tiny dot disappear? But so did the cursor in the lower left corner of the screen. In fact, if you type anything at this time, you will not see what you are typing. Yet what you type is still going into the computer's memory. In order to see that the computer is still all right and will still respond to you, type

```
HI HI-RES 2 {RETURN}
```

You still cannot see anything, but the computer should have beeped at you for typing such nonsense. (Remember, the computer does not understand vocabulary from the English language, it only understands vocabulary from its built-in BASIC language.) To see some information on this fourth screen, type very carefully

```
HCOLOR= 3 {RETURN}
HPLOT 140,96 {RETURN}
```

If you have typed accurately and remembered to press the RETURN key after each line, you should see a small dot approximately in the middle of the screen. This dot may be difficult to get since it requires you to type these instructions without seeing what you are typing. If you make a mistake typing and do not get the dot to appear, press the RETURN key a few times and then retype the above instructions.

This dot in the center of your screen is on the Apple's fourth display screen—the High-Resolution Screen 2 or High-Resolution Page 2. In fact, we now have information on three of the four screens at the same time; our instructions are on the text screen or Page 1, the small dot in the upper left portion of the screen is on High-Resolution Page 1, and the small dot in the approximate center of the screen is on High-Resolution Page 2.

Moving Between Screens

The next step is to flip back to Page 1 (text) so that you can see what you type. Do you remember the word used to get back to full-screen text from the low-resolution mode of Page 1? Use that same BASIC word now to display text information instead of graphic information. Type

```
TEXT {RETURN}
```

Immediately, the small dot disappears and the text screen (Page 1) is restored to view. Now you can see all the things you may have tried while High-Resolution Screen 2 was displayed. Notice that this time we did not get the inverse @ signs and that the text information was retained by the computer when we were viewing HGR2.

What we have done is to switch channels. We have switched from watching Channel 1 (Text Page 1) to Channel 1H (High-Resolution Screen 1), then on to Channel 2H (High-Resolution Screen 2), and finally back to Channel 1 (Text Page 1) again. And, just as when you switch television channels you may find that the same program is on that was on earlier, you may find the same information, or at least some of it, still being displayed on the computer screen. This analogy can help explain why the word TEXT removes information from the screen at one time and returns you to a previous display at another time.

The first time you used the word TEXT, you were acting like an individual changing video tapes in a video-tape recorder. Just as only one video tape can be seen at a time on your television, only one type of display can be seen at a time on Text Page 1 (i.e., either text or low-resolution graphics or a combination of four lines of text and 20 lines of graphics). But when you used the word TEXT on High-Resolution 2 in order to return to Page 1, you were acting like an individual switching between channels (between areas of the computer's memory).

This explanation should become clearer after you have tried the following demonstration. Flipping or switching between pages or screens will give you visual affirmation of the existence of the three separate screens.

Begin by clearing the three screens to get them ready for new information. The first step is to clear Text Page 1. Type

```
HOME {RETURN}
```

The screen should be blank, and the cursor should be in the upper left corner. The second step is to clear High-Resolution Screen 1. Type

```
VTAB 22 {RETURN}
HGR {RETURN}
```

The VTAB 22 statement is necessary in order to move the cursor down to one of the four text lines that are still visible after issuing the HGR command. The HGR instruction should again clear the screen, but this time the cursor will be in the lower left-hand corner. Finally we clear the second high-resolution screen. Type

```
HGR2 {RETURN}
```

There is not much you can do to make the cursor visible after issuing the HGR2 instruction, because part of the instruction informs the computer that you want to devote the entire screen to high-resolution graphics.

Now you have cleared the three most frequently used screens of information. Well, not quite – the HGR and HGR2 instructions are on the text screen though we cannot see them. Next you are going to "flip" between these three pages to see that they really can contain different information at the same time. The instructions you need to do this vary according to the screen you want to see and the screen you currently are viewing. First, get back to the text screen by typing

```
TEXT {RETURN}
```

Issuing the TEXT instruction while one of the high-resolution screens is displayed does not destroy information on Text Page 1 unless there are low-resolution graphics on it. Also, the TEXT instruction always returns you to Text Page 1. As stated before, different instructions must be used to view Text Page 2, since that is the same area of memory used to store instructions in an Applesoft program.

Your next task will be to go to High-Resolution Screen 1 and place information on that screen. Type

```
HGR {RETURN}
HCOLOR = 3 {RETURN}
HPLOT 0,80 TO 279,80 {RETURN}
```

These instructions

- clear the screen (HGR)
- set the color equal to white (HCOLOR = 3)
- draw a horizontal line from the left side of the screen to the right side approximately halfway down. 0,80 is a point 80 rows down the far left side of the screen; 279,80 is a point 80 rows down the far right side of the screen.

There are 280 horizontal points on each row (numbered 0 through 279) and 160 rows (numbered 0 through 159) on the first high-resolution screen. If you multiply 280 points across by 160 points down, you get a total of 44,800 points that can be accessed on the first high-resolution screen after issuing an HGR instruction. There are additional points that can be used and displayed, but for now I will limit the discussion to these numbers.

Placing Information on Several Screens

The next problem will be to use and display information on the second high-resolution screen while we still have information on the other two screens. Type

```
HGR2 {RETURN}
HCOLOR = 6 {RETURN}
HPLOT 140,0 TO 140,191 {RETURN}
```

Type carefully, since you cannot see what you are typing. These instructions clear the screen, set the color to blue and draw a vertical line from the top of the screen to the bottom approximately half-way across. There are still 280 horizontal points on each row, but now there are 192 rows (numbered 0 through 191), for a total of 53,760 points that can be accessed on the second high-resolution screen after issuing an HGR2 instruction. Since HR2 does not normally have the four lines of text, an HGR2 instruction makes use of that area for graphics instead of text. You will eventually see that all 53,760 points can also be accessed on High-Resolution Screen 1.

Now you have information on all three screens at the same time. You can view that information by flipping between pages. You should have a blue vertical line on High-Res Page 2, a white horizontal line on Hi-Res Page 1, and a series of instructions on Text Page 1. If you make a mistake typing or have trouble getting the indicated display, simply press the RETURN key and type the instruction again. It is difficult to type instructions when you cannot see what you are typing, but at this point, this is the best method to achieve our purposes. The BASIC instruction POKE is used to place some value directly into the computer's memory, thus altering the way the computer operates. Type

```
POKE-16300,0 {RETURN}
```

The display should be Hi-Res Page 1. Type

```
POKE -16303,0 {RETURN}
```

The display should now be the text page. Type

```
POKE -16304,0 {RETURN}
```

Back to the horizontal line (Hi-Res Page 1). Type

```
POKE -16299,0 {RETURN}
```

Back to the vertical line on Hi-Res Page 2. Type

```
TEXT {RETURN}
```

At last, you are back on the text page, where you can see what you are typing. Remember that the Text Page 1 can also display low-resolution graphics (either in place of text or along with four lines of text) so that the page flipping we have just done could also have included low-resolution graphics. The purpose at this time is not to introduce all the different switches that can be set in different ways for different combinations of displays, but to demonstrate the existence of three of the four display screens and the fact that, because these screens are separate, they can contain different information at the same time. Please do not worry if you have not followed everything. All you really need to understand is that they are separate screens, existing for different purposes.

The Deferred Mode

All of the steps you have taken so far have been taken in what is called the **immediate mode.** In this mode, the instructions you type are acted upon as soon as the RETURN key is pressed. The immediate mode, sometimes called the direct mode, can be very helpful in determining what errors exist in your programs. Most of the work you will be doing, however, will be in what is called the **deferred mode.**

In the deferred mode, sometimes called the indirect mode, the computer does not follow the instructions immediately, but waits until it is told to RUN the program (list of commands or instructions). It defers action until it is specifically told to act. Line numbers tell the computer the exact sequence in which it is to follow the program instructions.

The following program contains the instructions given for placing information on the three common screens and then viewing or flipping between those screens without destroying the information on them.

Type (remember to press the {RETURN} key after each line)

```
100 TEXT
110 HGR
120 HCOLOR = 3
130 HPLOT 8,80 TO 279,80
140 HGR2
150 HCOLOR = 6
160 HPLOT 140,0 TO 140,191
170 POKE - 16300,0
180 POKE - 16303,0
190 POKE - 16304,0
```

```
200 POKE - 16299,0
210 TEXT
```

The line numbers used are arbitrary. I like to begin most programs with line number 100 so the line numbers are positioned nicely.

It is also a good idea to leave room between line numbers so you can include additional instructions if necessary. In fact, when you RUN the program, you will see that you need more instructions to slow the computer down to be able to actually see the different screens. To verify the need for additional instructions, type

```
RUN {RETURN}
```

and watch carefully! The computer responds very quickly and the screens go by too fast for you to be able to really see them.

When you used these instructions in the immediate mode, the computer had to wait for the next instruction to be typed in from the keyboard. While it waited, you had a chance to view the different screens and information. Under program control, the computer does not wait unless specifically told to do so. It processes each instruction immediately after the preceding instruction has been completed. Thus, it is necessary to provide additional instructions to tell the computer to wait until you are ready to proceed.

There are a number of different options available at this point. One very quick method would be to add null or empty input statements between certain instructions. In order to see the effect of this method, type (remember the {RETURN} key after each line)

```
135 INPUT NUL$
165 INPUT NUL$
175 INPUT NUL$
185 INPUT NUL$
195 INPUT NUL$
205 INPUT NUL$
```

After you have finished adding these instructions, type

```
RUN {RETURN}
```

The computer displays a horizontal white line and a ? in the lower left corner. If you continue to press the RETURN key, you will eventually flip through all the instructions given above until you get back to the text page and the blinking cursor. But at points, you may not be certain which screen you are actually viewing.

A better method, to slow the computer according to your preference, is the method presented in the program at the end of this chapter. The only difference between this program and the one given above is in the instructions that

1. indicate to the computer when the user is ready to see the different screens
2. tell what screens the user is viewing or going to view

Without these user instructions, the computer executes all the commands too quickly for you to see the different screens and displays the screens but does not display a message saying which screen is being shown. A similar tutorial program is included at the end of this chapter that covers the instructions and procedures presented in Chapter 2.

In future chapters you will begin creating programs, or series of instructions, to inform the computer of a desired action. At this point, do not worry if you don't understand all the instructions you are typing into the computer. Eventually they will all be explained.

Please read over the listings for both programs at the end of this chapter. They are heavily documented with remarks explaining what the various instructions tell the computer to do. One of the best ways to learn programming is to examine a fully documented program and understand what the programmer was instructing the computer to do.

In the next chapter we will look at the IIGS display modes. If you are not using a IIGS you might want to skip to Chapter 5, which will examine the differences between low-resolution graphics and high-resolution graphics and between the instructions that are used with each.

REVIEW QUESTIONS

1. High-Resolution Page 1 is reached through the Applesoft BASIC reserved word ______________.
2. High-Resolution Page 2 is reached through the Applesoft BASIC reserved word ______________.
3. Which instruction is used to set the color in high-resolution?
4. Which instruction is used to display a single point on the high-resolution screens?
5. True or False? High-resolution graphic display destroys information on the text page.
6. True or False? Normally after an HGR2 command, you are able to see the instructions you type into the computer.
7. The command HOME does what?
8. How many horizontal points are there on the high-resolution screens?
9. How many vertical points are there on High-Resolution Page 1?
10. How many vertical points are there on High Resolution Page 2?
11. What reserved word is used to directly place values into the computer's memory?
12. When the computer responds immediately to commands it is operating in the ______________ mode.
13. When the computer waits for a specific instruction before executing instructions, it is operating in the ______________ mode.
14. What reserved word is used to inform the computer that it should begin execution of an Applesoft BASIC program?

Chapter 2 Tutorial: Text Page 1

```
100  REM ***--CHAP.2.TUT--***
110 :
120 :
130  REM **--TEXT SCREEN OR PAGE 1--**
140  TEXT : REM MAKE SURE DISPLAY IN TEXT MODE
150  HOME : REM CLEAR ENTIRE SCREEN--CURSOR UPPER LEFT
160  VTAB 3: REM VERTICAL TAB 3 LINES DOWN SCREEN
170  INVERSE
180  PRINT "TEXT SCREEN/PAGE 1"
190  NORMAL
200  VTAB 10: REM  VERTICAL TAB 10 LINES DOWN SCREEN
210  PRINT "]PRINT 'HELLO, I AM AN APPLE COMPUTER.'"
220  PRINT "HELLO, I AM AN APPLE COMPUTER."
230  GOSUB 2000: REM  RETURN KEY ROUTINE
240 :
250 :
260  REM  **--LOW RESOLUTION SCREEN/PAGE 1--**
270  HOME : REM  CLEAR ENTIRE SCREEN
280  INVERSE
290  VTAB 3: REM  VERTICAL TAB
300  PRINT "LOW RESOLUTION SCREEN/PAGE 1 COMMANDS"
310  NORMAL
320  VTAB 10: REM  VERTICAL TAB
330  PRINT "GR"
340  PRINT
350  PRINT "COLOR = 6"
360  PRINT
370  PRINT "PLOT 20,20"
380  GOSUB 2000: REM  RETURN KEY ROUTINE
390  GR : REM  INITIATE LOW-RES MODE
400  COLOR= 6: REM  SET LOW-RES COLOR TO BLUE
410  PLOT 20,20: REM  DISPLAY SINGLE BLUE POINT
420  GOSUB 2000: REM  RETURN KEY ROUTINE
430 :
440 :
450  REM  **--RETURN TO TEXT SCREEN--**
460  VTAB 21: REM  VERTICAL TAB
470  PRINT "TEXT COMMAND RETURNS US TO TEXT SCREEN"
480  GOSUB 2000: REM  RETURN KEY ROUTINE
490  TEXT : REM  SET DISPLAY MODE TO TEXT
500  VTAB 22: REM  VERTICAL TAB
510  CALL  - 868: REM  CLEAR LINE
520  GOSUB 2000: REM  RETURN KEY ROUTINE
530 :
540 :
```

```
550  REM **--CLEAR TEXT SCREEN/HOME CURSOR--**
560  HOME : REM  CLEAR ENTIRE SCREEN
570  VTAB 3: REM  VERTICAL TAB
580  INVERSE
590  PRINT "TEXT SCREEN/PAGE 1"
600  NORMAL
610  VTAB 10: REM  VERTICAL TAB
620  PRINT "HOME"
630  GOSUB 2000: REM  RETURN KEY ROUTINE
640  HOME : REM  CLEAR ENTIRE SCREEN
650  VTAB 10: REM  VERTICAL TAB
660  INVERSE
670  PRINT "END OF CHAPTER 2 TUTORIAL"
680  NORMAL
690 :
700 :
710  REM  **--END--**
720  END
730 :
740 :
2000  REM  **--RETURN KEY ROUTINE--**
2010  VTAB 23: REM  VERTICAL TAB
2020  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY TO CONTINUE:";L$
2030  RETURN
```

Chapter 3 Tutorial: High-Resolution Page 1

```
100  REM ***--CHAP.3.TUT--***
110 :
120 :
130  REM  **--TEXT PAGE--**
140  TEXT : REM  MAKE SURE DISPLAY IN TEXT MODE
150  HOME : REM  CLEAR ENTIRE SCREEN--CURSOR UPPER LEFT
160  VTAB 10: REM  VERTICAL TAB 10 LINES DOWN SCREEN
170  PRINT "THIS IS THE TEXT SCREEN OR TEXT PAGE."
180  GOSUB 2000: REM  RETURN KEY ROUTINE
190 :
200 :
210  REM  **--HI-RES PAGE 1--**
220  HGR : REM  INITIATE HI-RES PAGE 1
230  HCOLOR= 3: REM  SET HI-RES COLOR TO WHITE
240  HPLOT 0,80 TO 279,80: REM  DRAW HORIZ.  WHITE LINE
250  VTAB 21: REM  VERTICAL TAB
260  PRINT "AFTER THE HGR INSTRUCTION, THE TOP OF"
270  PRINT "THE SCREEN DISPLAYS HI-RES PAGE 1."
280  PRINT "THE LAST 4 LINES CAN DISPLAY TEXT."
290  GOSUB 2000: REM  RETURN KEY ROUTINE
300  VTAB 21: REM  VERTICAL TAB
310  CALL  - 868: REM  CLEAR LINE
320 :
330 :
340  REM   **--HI-RES PAGE 2--**
350  REM  GIVE INFO.NOW! HGR2 DOESN'T ALLOW TEXT & GRAPHICS
360  PRINT "THE FOLL.SCREEN WILL BE HI-RES PAGE 2."
370  PRINT "THE HGR2 INSTRUCTION DOES NOT GIVE 4"
380  PRINT "LINES OF TEXT. REM.'RETURN' TO GO ON!"
390  GOSUB 2000: REM  RETURN KEY ROUTINE
400  HGR2 : REM  INITIATE HI-RES PAGE 2
410  HCOLOR= 6: REM  SET HI-RES COLOR TO BLUE
420  HPLOT 140,0 TO 140,191: REM  DRAW VERTICAL BLUE LINE
430  GOSUB 2000: REM  RETURN KEY ROUTINE
440 :
450 :
460  REM **--BACK TO HI-RES PG.1--**
470  POKE  - 16301,0: REM  SWITCH FROM FULL GRAPH.TO TEXT/GRAPH
480  POKE  - 16300,0: REM  SWITCH DISPLAY FROM PG.2 TO PG.1
490  VTAB 21: REM  VERTICAL TAB
500  CALL  - 868: REM  CLEAR LINE
510  PRINT "BACK TO HI-RES PAGE 1 & HORIZ. LINE."
520  GOSUB 2000: REM  RETURN KEY ROUTINE
530 :
540 :
550  REM **--BACK TO TEXT PAGE--**
```

```
560  VTAB 15: REM  VERTICAL TAB
570  CALL  - 958: REM  CLEAR TO BOTTOM OF SCREEN
580  POKE  - 16303,0: REM  SWITCH DISPLAY FROM GRAPHICS TO TEXT
590  VTAB 21: REM  VERTICAL TAB
600  PRINT "BACK TO THE TEXT PAGE."
610  GOSUB 2000: REM  RETURN KEY ROUTINE
620 :
630 :
640  REM **--BACK TO HI-RES PG.1--**
650  POKE  - 16304,0: REM  SWITCH DISPLAY FROM TEXT TO GRAPHICS
660  VTAB 21: REM  VERTICAL TAB
670  CALL  - 868: REM  CLEAR LINE
680  PRINT "BACK AGAIN TO HI-RES PAGE 1."
690  GOSUB 2000: REM  RETURN KEY ROUTINE
700 :
710 :
720  REM **--BACK TO HI-RES PG.2--**
730  VTAB 15: REM  VERTICAL TAB
740  CALL  - 958: REM  CLEAR TO BOTTOM OF SCREEN
750  VTAB 21: REM  VERTICAL TAB
760  PRINT "THE FOLL. SCREEN WILL BE HI-RES. PG.2:"
770  PRINT "REMEMBER TO PRESS 'RETURN' TO GO ON."
780  GOSUB 2000: REM  RETURN KEY ROUTINE
790  POKE  - 16302,0: REM  SWITCH DISPLAY FROM TEXT/GRAPH.TO FULL GRAPH.
800  POKE  - 16299,0: REM  SWITCH DISPLAY FROM PG.1 TO PG.2
810  GOSUB 2000: REM  RETURN KEY ROUTINE
820 :
830 :
840  REM **--TEXT PAGE FOR FINAL INFO.--**
850  TEXT : REM  SWITCH DISPLAY FROM GRAPHICS TO TEXT
860  HOME : REM  CLEAR ENTIRE SCREEN
870  VTAB 10: REM  VERTICAL TAB
880  PRINT "FINALLY BACK TO THE TEXT SCREEN!"
890  GOSUB 2000: REM  RETURN KEY ROUTINE
900  HOME : REM  CLEAR ENTIRE SCREEN
910  VTAB 5: REM  VERTICAL TAB
920  PRINT "THIS HAS BEEN A DEMONSTRATION OF THE"
930  PRINT
940  PRINT "COMPUTER'S ABILITY TO RETAIN AND "
950  PRINT
960  PRINT "ALTERNATELY DISPLAY DIFFERENT SCREENS"
970  PRINT
980  PRINT "WITH TEXT, GRAPHIC, OR A COMBINATION"
990  PRINT
1000  PRINT "OF BOTH TEXT AND GRAPHIC INFORMATION."
1010  PRINT
1020  INVERSE
1030  PRINT "ALL FINISHED"
```

```
1040  NORMAL
1050  END
1060 :
1070 :
2000  REM  **--RETURN KEY ROUTINE--**
2010  VTAB 24: REM  VERTICAL TAB
2020  CALL  - 868: REM  CLEAR LINE
2030  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY TO CONTINUE:";L$
2040  VTAB 23: REM  VERTICAL TAB
2050  CALL  - 868: REM  CLEAR LINE
2060  RETURN
```

CHAPTER 4

IIGS Display Modes

Although the IIGS is an Apple *II* computer, it is far more than an improved version. In reality, the IIGS is a new computer that contains an Apple *II* as just one part of the machine. The Apple IIGS is sufficiently advanced that many of the limitations that exist for various reasons in the Apple *II* computers do not need to exist in the IIGS. Yet, to maintain software and hardware compatibility, those limitations continue. In order to remain compatible with existing Apple *II* computers and software, yet allow users to take full advantage of the capabilities of the IIGS, Apple created two modes of operation. These two modes of operation are known by different names. Some refer to the two modes as "8-bit" and "16-bit". Others refer to them as "classic" and "native," "*II*e/c" and "IIGS" or "ProDOS 8" and "ProDOS16." I will refer to the two modes as the 8-bit mode and the 16-bit mode, since these names are somewhat descriptive.

The main chip (the CPU or central processing unit) on the old Apple *II* line of computers could handle eight pieces of information at one time—something like a robot with eight arms. The main chip on the IIGS can deal with 16 pieces of information at one time—something like a robot with 16 arms. The robot analogy helps explain why programs created on the IIGS in 16-bit mode won't work on the older Apple *II* computers, yet programs created on the older Apple *II* computers will work on the IIGS. If you have a task that requires a robot to use 16 arms at one time, an eight-arm robot will not be able to do the job. Yet tasks requiring only eight arms can be done by the 16-arm robot. Therefore, programs that use the full capability of the IIGS *will not* work on the old Apple *II* computers because those computers don't have enough "arms" to get the job done. Programs that work on the older Apple *II*'s *will* run on the IIGS, but will not make use of the machine's additional capabilities (the other eight arms in the robot analogy). Applesoft BASIC is a program that works both on the IIGS and on older Apple *II*'s.

Usually we don't think of Applesoft as a program but as a language used to create programs. In reality, Applesoft is just another program as far as the computer is concerned. The Applesoft BASIC program was created to work in the world of 8-bit computers. If Applesoft were modified so that it took advantage of the 16-bit capabilities of the IIGS,

programs created with the new Applesoft would not work on older Apple *II*'s. Since the program was never modified, programs created with Applesoft will function on all Apple *II* machines, though these programs supposedly do not take advantage of the full capability of the IIGS computer.

I intentionally used the word "supposedly." Applesoft was created with enough flexibility that persistent programmers can make use of many of the new capabilities of the IIGS. We will cover some of the routines that you can use to access these capabilities later, but for now you should understand that, generally speaking, Applesoft works in the 8-bit mode, not the 16-bit mode.

APPLESOFT BACKGROUND

This section includes background information that will help you understand why Applesoft BASIC cannot directly and easily access all the capabilities of the IIGS.

The fact that Applesoft has not changed makes it possible to write a BASIC program on any Apple *II* computer and know that it will run on all versions of the Apple *II*, but it also means that programmers using BASIC must go through ridiculous gyrations to use the new features of the IIGS. These programming feats would not be necessary if Apple had made Applesoft upward compatible instead of downward compatible. In other words, Apple could have created a 16-bit version of Applesoft that was capable of running 8-bit programs, yet included additional reserved words that would access the new features (sound and graphics especially) found on the IIGS. Programs created on a IIGS that made use of these new features would not have run on older Apple *II*'s, but programs that did not use the new features would have been compatible with 8-bit Applesoft and would work on all versions of the Apple *II*. Apple used this upward compatible but not downward compatible procedure for the ProDOS operating system.

When Apple first introduced the Macintosh computer, they did not officially bring out a version of BASIC with it. Microsoft eventually offered a decent version of BASIC that is capable of accessing most of the capabilities of the Macintosh. Apple, apparently, has given up on ever supplying its own version of BASIC for the Macintosh, because the company expects that serious software developers will use languages other than BASIC.

This attitude overlooks the fact that Apple owes its share of the educational market to the success of Applesoft in accessing the capabilities of the Apple *II*. Applesoft has allowed thousands of people to create tens of thousands of programs, commercial and noncommercial, that work on any Apple *II* computer. Many of these programs will not be upgraded to take advantage of the capabilities of the IIGS because upgrading would mean rewriting them in a new language. Many of the thousands of people who struggled to learn BASIC are not about to spend the time necessary to learn each new language that comes along.

At one time or another Assembly language, Pascal, Logo, FORTH, and Plato were languages that were *the* language to learn and program in—especially for educators. Today

the new language that is a must to learn and use is C, but Applesoft BASIC will still be around when the next trendy language supplants C.

BASIC's greatest weakness is in fact its strength. It is flexible. It is a tool, not a fixed machine. It does not require genius to use yet allows genius to produce. So, in spite of Apple's decision not to upgrade Applesoft, the flexibility of Applesoft makes it possible (in combination with some assembly-language routines) to access some of the capabilities of the IIGS.

IIGS DISPLAY MODES

Applesoft BASIC can easily access some of the IIGS display modes. This section describes all of the additional display modes available on a IIGS whether or not they can be accessed using BASIC. Chapters 9 and 10 will explain BASIC programming techniques used to access the various display modes on the IIGS.

The IIGS has the standard Apple *II* display modes of 40-column Text/Low Res 1, 40-column Text/Low-Res 2, High-Res 1, and High-Res 2. All Four of these display modes are directly supported by Applesoft reserved words. The IIGS also includes the standard *II*e and *II*c displays of 80-column text, double low- (or medium-) resolution, and double high-resolution. These three display modes are not directly supported by Applesoft reserved words. Eighty-column text can be accessed through the PR#3 command (directing output to Port #3), but double low-resolution and double high-resolution must be accessed through various calls to assembly-language routines.

The new super high-resolution display modes on the IIGS are not directly supported by Applesoft, nor have standard calls and/or routines been developed to allow easy access to them. Standardization has not occurred because the IIGS has not been available for long and because the IIGS is a far more complex machine than the other Apple *II*'s, with vastly increased capability. Additionally, Apple has instituted the Macintosh User Interface as the standard to be used on the IIGS. Applesoft does not support the Mac User Interface: the mouse, windows, menu bar, dialog box, scrolling, events, etc. The tool box and these other capabilities must also be accessed through calls, peeks, pokes or assembly-language routines.

Finally, the IIGS has introduced colored text to the Apple *II* line of computers. Once again, Applesoft does not have a direct method of setting the background, border or text color. These additions must be done through the "back door," with assembly-language routines or calls to certain established locations with certain parameters already set.

The capability of the IIGS is extraordinary, but access to its capability through Applesoft is very difficult. One of the purposes of this book is to provide a method that will make better use of the IIGS capability through Applesoft. This approach is not unique. Other computer manufacturers have extended their versions of BASIC to provide additional capability. I believe that extensions to Applesoft BASIC will eventually allow easier access to the various capabilities of the IIGS.

Additional IIGS Display Modes

Some of the additional display modes on the IIGS require specific display devices. For example, 80-column colored text can only be read on RGB color monitors, not on composite color monitors.

There are two additional display modes available on the Apple IIGS. Both super high-resolution modes operate quite differently than the other display modes do. Super High-Resolution 1 has a resolution of 320 horizontal points on each line and 200 vertical lines. Super High-Resolution 2 has a resolution of 640 horizontal points on each line and the same 200 vertical lines. Yet, both modes use the same space in memory and the same amount of memory.

Remember that High-Resolution Page 1 (HGR) has its own memory space (hex 2000 to 3FFF) and High-Resolution Page 2 has a separate memory location (hex 4000 to 5FFF). But both super high-resolution modes exist in the same memory location ($E1/2000 to $E1/9FFF). In fact, structured properly, both modes can exist on screen at the same time. Each horizontal line requires 160 bytes (spaces) of memory. If the computer has been told that the line is in 320 mode, each dot takes up half a byte (space) of memory (160 bytes $\times$ 2 = 320 points). If the line is in 640 mode then each dot requires one fourth of a byte (160 bytes $\times$ 4 = 640 points). In other words, the more dots or points on each line, the smaller those dots will be—and the finer the resolution.

There are other differences between the two modes. 320 mode, theoretically, allows more colors than does 640 mode. (I say "theoretically" because clever programmers can work the 640 mode so they get more colors than the manuals say is possible.) In 320 mode, you can display any color from any of the 16 possible color tables at any point on the screen. In 640 mode, you are somewhat restricted in the number of colors that you can display at any point.

The 320 mode actually consists of two different modes—regular mode and fill mode. *Fill mode is not available in 640 super-resolution.* Fill mode allows you to fill a space with a color you specify. In other words, you define an area, such as a circle, and then inform the computer that the area should be filled with a certain color. Fill mode is very useful when you are creating many screens containing numerous shapes that must be drawn quickly in different colors (i.e., animation or certain types of charts). In fact, it appears that many of the early programs specifically written for the IIGS use the 320 mode rather than the 640 mode so they can take advantage of the fill option available in the 320 super high-resolution display mode.

SUPER HIGH-RESOLUTION TUTORIAL

As stated, one of the purposes of this book is to give you access to some of the capabilities of the IIGS through the built-in language—Applesoft BASIC. Because access is not direct

and easy, a simple tutorial demonstrating super-high resolution is not possible. Instead, I will simply explain how you can view one of the super high-resolution screen from BASIC and then switch your computer back to BASIC.

In order to be certain that this tutorial will work, you must start up your IIGS with the Apple IIGS System Disk—not the Apple IIGS Tour Disk or any other disk. After inserting the correct diskette into the drive and turning on the computer, select BASIC.SYSTEM when you are asked to select the file you want to open. You can select this file by using either the mouse or the arrow keys. (Remember to press the key marked return after making your selection.) After pressing the RETURN key, the screen will clear and eventually you will see

```
PRODOS BASIC 1.1
COPYRIGHT APPLE, 1983—84
]
```

The wording and numbers may be slightly different on future releases of ProDOS, but the idea is that you are now looking at Text/Low-Resolution screen 1 in Applesoft BASIC. Because the System Disk's start-up program contains all the instructions necessary to display information on the super high-resolution screen, we can "flip the switch" and easily see what the super high-resolution screen area looks like. One caution is necessary. Once you have switched the display to super high-resolution, you will not be able to see anything you type. Don't worry though—what you type will still provide instructions to the computer. Therefore, if you do not get the stated results, simply press the RETURN key and try again.

The first step is to switch from Text/Low-Resolution 1 to Super High-Resolution. In order to do this type

```
POKE 49193,163 {RETURN}
```

Once you have pressed the RETURN key, you should see much the same display that you did when you first started up the computer with the System Disk. The System Disk's start-up program is written in 16-bit mode and makes use of the super high-resolution capabilities of the IIGS to present the user with the choices available. The choices themselves are not visible, because they are part of a dialog-box routine that can be displayed on the super high-resolution screens. Recent releases of the System Disk do not include the dialog box. The mouse arrow is visible in the upper left-hand corner of the screen but it too is not active, because all we have done is to activate the display of the super high-resolution screen. We have not activated the dialog-box routine or any of the mouse routines. The instruction "POKE 49193,163" simply tells the computer to switch the display over to the super high-resolution mode. In order to do anything on this screen, additional instructions are necessary. Some of those instructions will be covered in Chapters 10 and 11.

To return the Text/Low Resolution Screen 1, type the following (even though you cannot see what you are typing the computer is still reading the keyboard and will act on your instructions):

```
POKE 49193,1 {RETURN}
```

If you have typed this instruction carefully and accurately, the computer will switch back to the Text/Low-Resolution Screen 1 for its display. Placing the decimal value 163 into memory location 49193 informs the computer that the super high-resolution memory area is to be used for the screen display. Placing the decimal value 1 into that memory location informs the computer that the display is to be returned to normal Apple *II* (8-bit) operation. To conclude this chapter, place the computer into different display modes (such as low-resolution graphics or high-resolution graphics) and then try to switch to the super high-resolution screen and back. You cannot hurt the computer by switching display screens.

REVIEW QUESTIONS

1. What are the two modes of operation for the IIGS?
2. Name the two super high-resolution display modes.
3. How many vertical lines are available in each super high-resolution mode?
4. True or False? Each super high-resolution mode has its own memory location just as High-Res 1 and High-Res 2 do.
5. True or False? Fill mode is not available in 640 super high-resolution.

CHAPTER 5

Differences Between Low- and High-Resolution Screens

This chapter will cover the major differences between the two most common graphic modes available on the Apple: low-resolution and high-resolution. Both of these graphic modes are directly supported by Applesoft commands. Later chapters will cover each mode individually and compare them to the other graphic modes available on later Apple *II* computers.

Most of the material I have seen on Apple graphics covers the low-resolution and/or high-resolution modes but does not provide a clear comparison of the differences between them. I believe it is important to understand what these differences are so you can make an intelligent choice regarding which to use for a particular application.

The first main area of difference between the low- and high-resolution screens is in the number of points on the screen (Table 5.1). Because low-resolution points are much larger than high-resolution points, low-resolution has only 40 columns (or vertical lines) on the screen, while high-resolution has 280 columns on the screen. These columns or lines are numbered beginning with the number 0, so that the last column in low-resolution is number 39, and the last column in high-resolution is number 279. This numbering system can be confusing at first, but it is really not much different than the numbering system on graph paper.

TABLE 5.1. Points on the Basic Apple Screens

	Low-Resolution		*High-Resolution*	
Full Screen				
Horizontal	40 Columns	(0–39)	280 Columns	(0–279)
Vertical	48 Rows	(0–47)	192 Rows	(0–191)
Mixed Screen (4 Lines Text/Graphics)				
Horizontal	40 Columns	(0–39)	280 Columns	(0–279)
Vertical	40 Rows	(0–39)	160 Rows	(0–159)

Low-resolution has 48 possible rows (or horizontal lines) while high-resolution has 192 possible rows. Again, the rows are numbered beginning with number 0 so that the last row is 47 for low-resolution and 191 for high-resolution. On both screens, the point 0,0 is at the upper left-hand corner.

These numbers of rows, 48 and 192, are the numbers for full-screen graphics. Often, graphic screens are not devoted entirely to graphics. Instead, a portion of the bottom half of the screen is used for four lines of text. Only 40 rows of graphic information are available in the mixed text/graphic mode on the low-resolution screen, while 160 rows of graphic information are available in mixed text/graphic mode on the high-resolution screen. Figures 5.1 through 5.8 illustrate the differences between low-resolution points and high-resolution points.

The second main area of difference between the two screens is in the availability of color (Table 5.2). Low-resolution has 16 colors, numbered 0 to 15, available at any point on the screen. High- resolution, according to the manuals, has six colors, numbered 0 to 7, with two numbers assigned to white (3 and 7) and two numbers assigned to black (0 and 4). Not every color is actually available at every point, however. Usually, odd-numbered colors are available in odd-numbered columns and even-numbered colors are available in even-numbered columns. There are exceptions to this statement, though. HCOLOR = 3 (white) will display a point in every column, but that point may not necessarily be white. Colors displayed in one column may be influenced by the color of an adjacent column. In other words, although you may have set the color properly and plotted your line correctly, if that line's color is influenced by the color of a line next to it (vertically), you may end up with an unexpected result. This adjacent-line problem usually occurs only with vertical lines. Horizontal lines normally plot out as you would expect them to. We will get into this problem in more detail in the chapters on high-resolution.

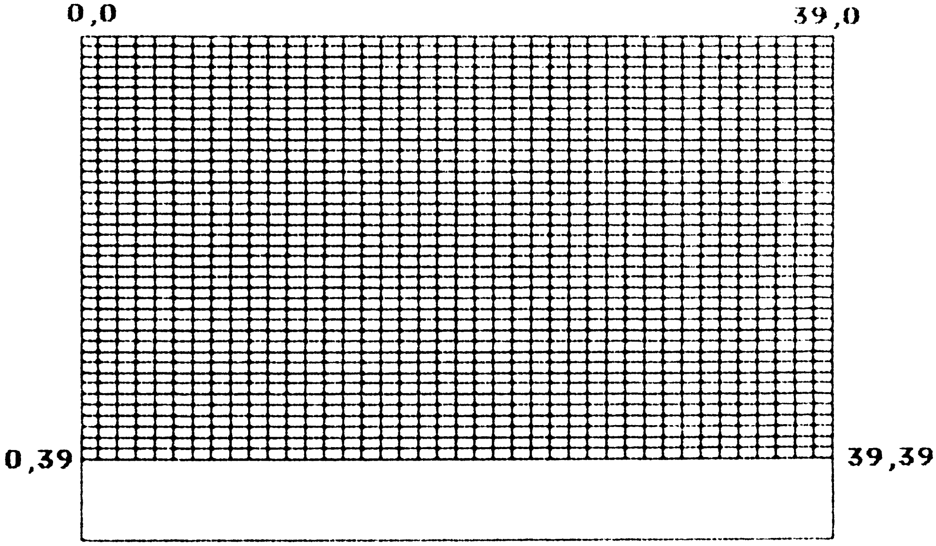


FIGURE 5.1. Low-resolution screen dimensions.

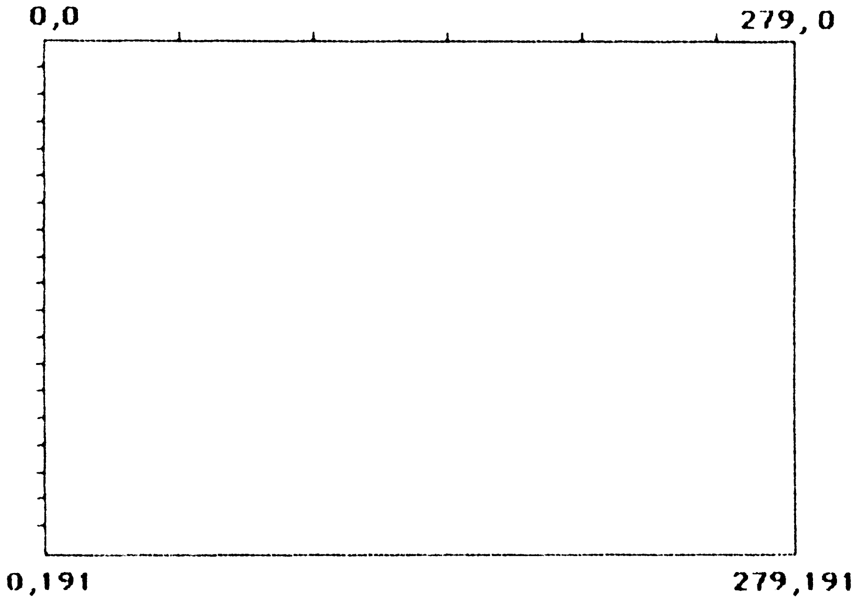


FIGURE 5.2. High-resolution screen dimensions.

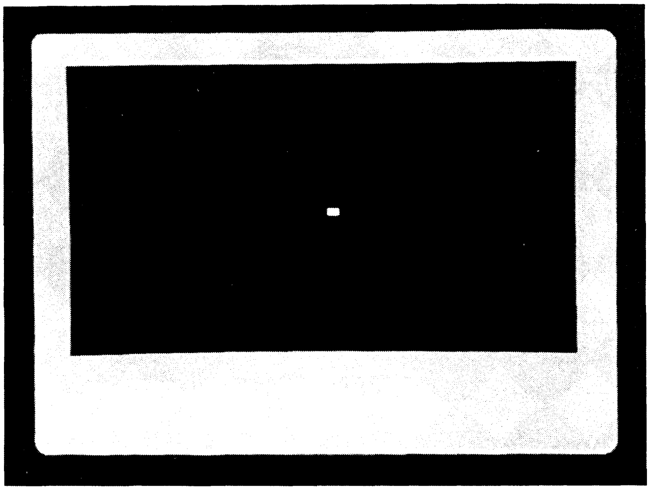


FIGURE 5.3. Single low-resolution point.

The third main area of difference between high- and low-resolution is in the instruction set necessary to display graphic information (Table 5.3). There are two separate instruction sets: one for low-resolution and one for high-resolution. The low-resolution set

TABLE 5.2. Colors on the Basic Apple Screens

Low-Resolution		*High-Resolution*	
Color	*Color #*	*Color*	*Color #*
Black	0	Black	0 & 4
Magenta	1		
Dark Blue	2		
Violet	3	Violet	2
Dark Green	4		
Grey-1	5		
Medium Blue	6	Blue	6
Light Blue	7		
Brown	8		
Orange	9	Orange	5
Grey-2	10		
Pink	11		
Green	12	Green	1
Yellow	13		
Aqua	14		
White	15	White	3 & 7

TABLE 5.3. Instructions for the Basic Apple Screens

Low-Resolution	*High-Resolution*
GR	HGR (HGR2)
COLOR	HCOLOR
PLOT	HPLOT
HLIN	HPLOT X1,Y TO X2,Y
VLIN	HPLOT X,Y1 TO X,Y2
SCRN	DRAW
	XDRAW
	ROT
	SCALE
	SHLOAD

contains six basic instructions, and the high-resolution set contains nine basic instructions. The low-resolution set includes only one instruction that has no comparable high-resolution instruction, but the high-resolution set has five instructions for working with shapes not available in low-resolution. Figures 5.9 through 5.11 illustrate some of the differences (albeit in black and white), between low-resolution and high-resolution colors.

Perhaps the biggest difference between low-resolution and high-resolution commands is in the instructions known as the SHAPE TABLE instructions. These instructions give you the ability to manipulate different shapes on the high-resolution screen. There are no comparable instructions or shape-manipulation routines available for low-resolution graphics. Some programmers have created their own shape-manipulation routines for low-resolution graphics, but this capability is not standard on Apple *II* computers.

The final difference between the two modes is in the common, if not universal, perception as to the usefulness of either resolution. Until recently, low-resolution graphics have been virtually ignored for serious use, although originally, they were the only real graphics available on the Apple. When high-resolution graphics became universally available (at the point at which the routines were completely documented and included in Applesoft), almost all graphics work switched over to the high-resolution screens for some very good reasons. In the process, low-resolution graphics have been ignored for some uses

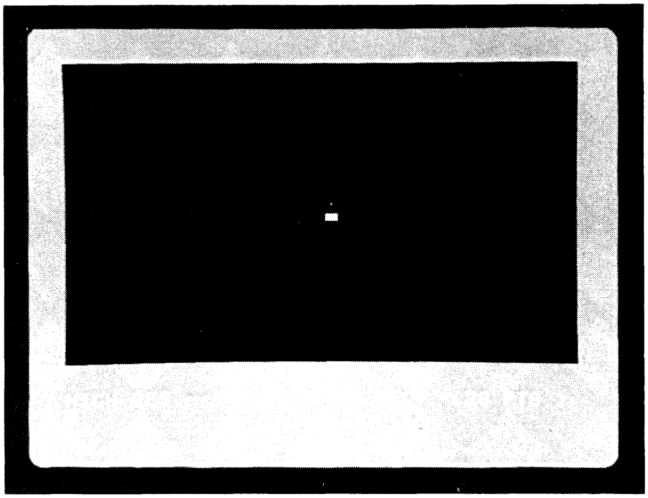


FIGURE 5.4. Single high-resolution point and block equal in size to a single low-resolution point.

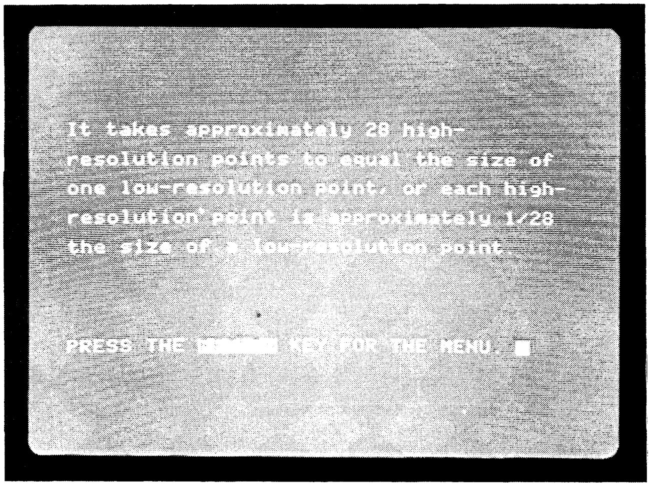


FIGURE 5.5. Single point comparison explanation.

for which they might be suitable. For example, most typing-practice programs use high-resolution graphics for their character display, although the larger letters of low-resolution graphics might provide beginners with an easier learning experience. This is especially true for younger learners. When you are writing a program you should be aware of the choices available and choose the resolution that best accomplishes your purpose.

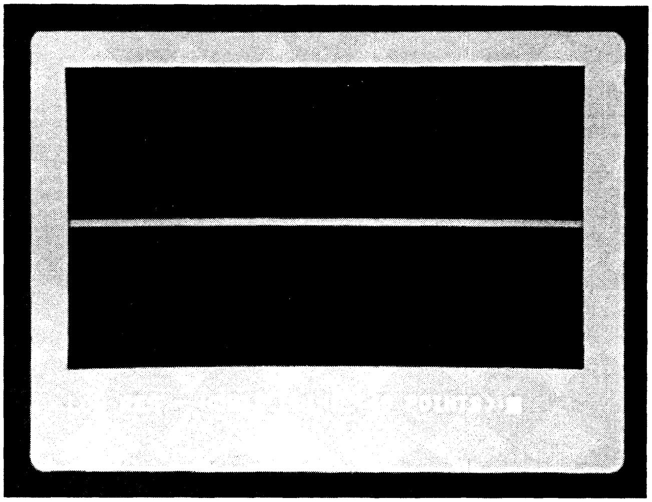


FIGURE 5.6. Low-resolution horizontal line.

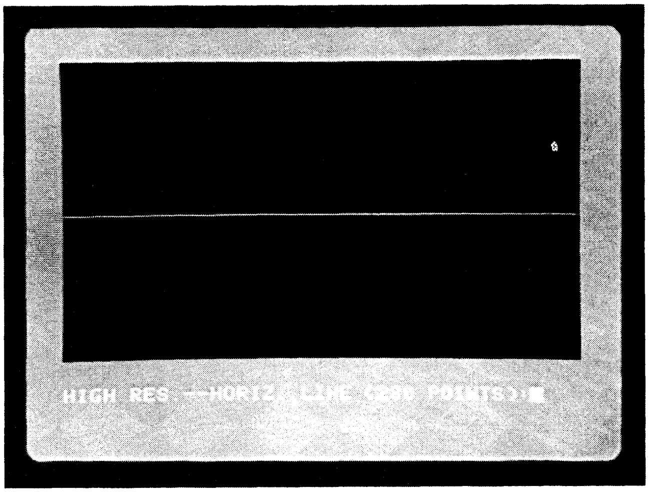


FIGURE 5.7. High-resolution horizontal line.

SCREEN-DISPLAY TUTORIAL

The tutorial for this chapter is illustrated in the programs listed at the end of the chapter. Each program uses the computer to teach you one of the differences between low-resolution and high-resolution graphics. A menu program has also been included. Once all the programs have been typed in you need only indicate the program number to see the comparison for that specific aspect. The RETURN key is not even necessary, except where indicated by the program's user instructions. All of the programs, with the exception of

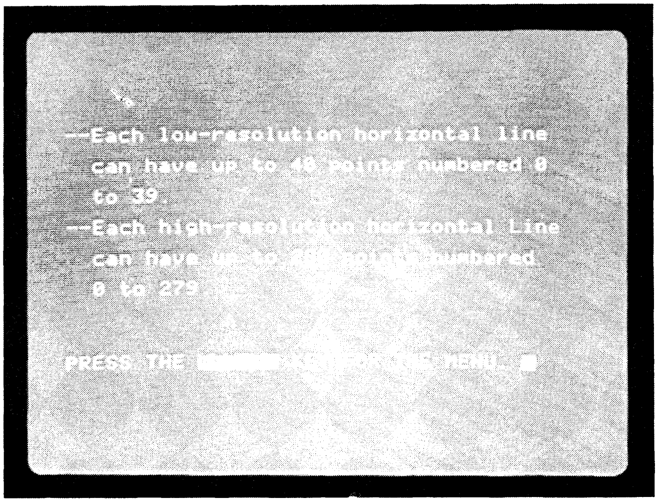


FIGURE 5.8. Horizontal line comparison explanation.

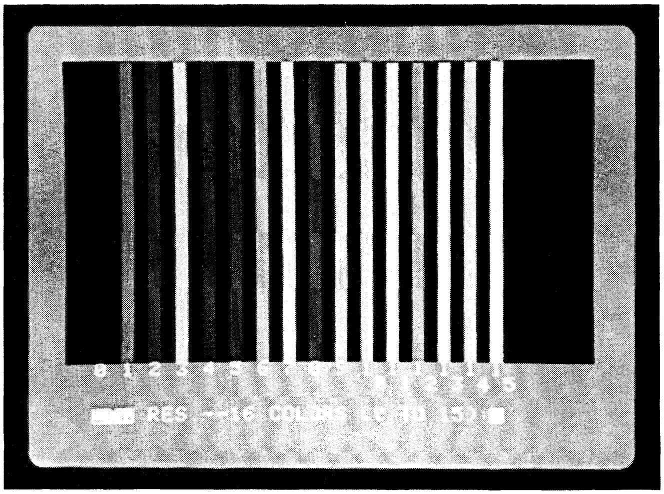


FIGURE 5.9. Low-resolution colors.

LOW.HIGH.RECT, execute with immediate response. The high-resolution portion of LOW.HIGH.RECT will take about two minutes and fifteen seconds to complete the high-resolution rectangle drawing (it will take about fifty seconds on a *II*GS running in FAST mode). Please let it finish even if you do not see something happen for a little while. As with all of the other programs, once the drawing is actually on the screen, there will be no delay in shifting or flipping between drawings or text on the different screens.

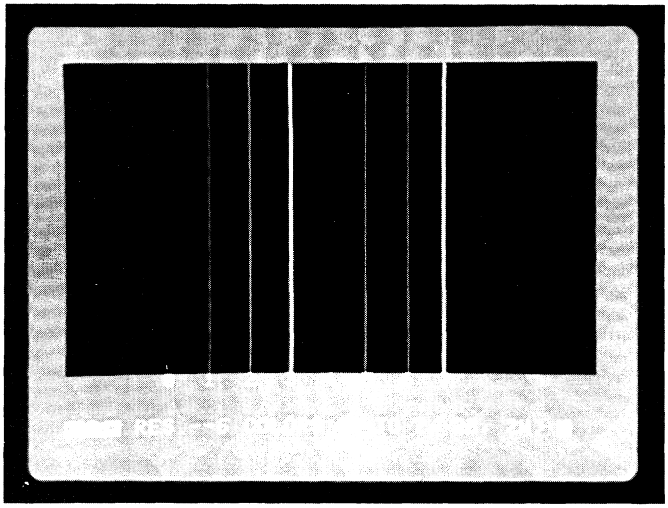


FIGURE 5.10. High-resolution colors.

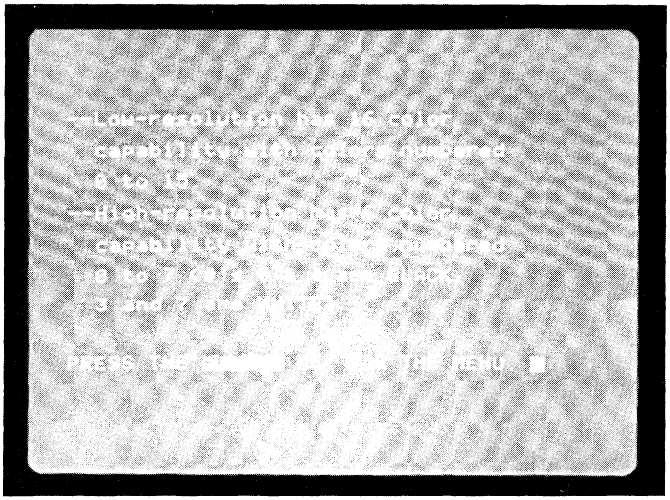


FIGURE 5.11. Color comparison explanation.

These programs involve a lot of typing and probably a lot of retyping. If you are not completely comfortable with the keyboard, all this typing will certainly improve your skills. But until you are comfortable with the keyboard, there are techniques that can reduce the time and errors involved in obtaining an operational series of comparison programs.

One technique is to key in the menu program and one other program within the series. Then, note and change only those lines that are different and require specific information for their particular display comparison. When all the changes have been made, save each

new program under the correct file name. In this way, you can quickly alter the instructions and save considerable typing. If you are fortunate enough to have a word-processing program that can handle BASIC listings, you can use that program to make the necessary changes.

One very important point remains. *Do not* attempt to put all of these programs together into one large program. As you will soon discover, certain precautions *must* be taken when you are using High-Resolution Page 1 or High-Resolution Page 2 with a program larger than about seven kilobytes of memory. Combining all these programs into one large program will exceed seven kilobytes of memory and cause you a great many problems.

Type carefully! The programs have been thoroughly tested and checked for errors. As listed in the book, these programs are fully operational under either DOS 3.3 or ProDOS. If you get an error message after you have finished typing in the programs and have tried to RUN them, the most likely problem is the typing, *not the program logic* or *any misprint of the listings*! I cannot emphasize enough the importance of careful typing and reading for mistyped instructions. If you do not want to spend the time keying in all these instructions, diskettes containing the programs are available. (Information on ordering the diskettes is given at the beginning of the book.)

REVIEW QUESTIONS

1. How many vertical columns do the high-resolution screens have?
2. How many vertical columns do the low-resolution screens have?
3. How many colors are directly possible in low-resolution graphics?
4. How many colors are directly possible in high-resolution graphics?
5. Which resolution has the most Applesoft graphics instructions?

Chapter 5 Tutorial Menu Program

```
100  REM  ***--LOW.HIGH.MENU--***
110 :
120 :
130 TB = 8
140 D$ =  CHR$ (4): REM  CONTROL D
150 :
160 :
170  REM  **--DISPLAY MENU--**
180  TEXT : HOME
190  VTAB 2
200  HTAB TB + 4
210  INVERSE
220  PRINT "LOW/HIGH MENU"
230  NORMAL
240  PRINT : PRINT
250  HTAB TB
260  PRINT "1.  SINGLE POINT COMPARISON"
270  PRINT : HTAB TB
280  PRINT "2.  HORIZONTAL LINE COMPARISON"
290  PRINT : HTAB TB
300  PRINT "3.  VERTICAL LINE COMPARISON"
310  PRINT : HTAB TB
320  PRINT "4.  COLOR COMPARISON"
330  PRINT : HTAB TB
340  PRINT "5.  CIRCLE COMPARISON"
350  PRINT : HTAB TB
360  PRINT "6.  RECTANGLE COMPARISON"
370  PRINT : HTAB TB
380  PRINT "7.  CATALOG"
390  PRINT : HTAB TB
400  PRINT "8.  END"
410  PRINT : HTAB TB
420  PRINT "WHICH NUMBER PLEASE? (";: INVERSE : PRINT "1-8";: NORMAL : PRINT
    "):";
430  GET NB$: PRINT
440 NB =  VAL (NB$)
450  IF NB < 1 OR NB > 8 THEN PRINT : HTAB TB: INVERSE : PRINT "INCORRE
    CT CHOICE!": NORMAL : GOTO 190
460  IF NB = 1 THEN 1000
470  IF NB = 2 THEN 2000
480  IF NB = 3 THEN 3000
490  IF NB = 4 THEN 4000
500  IF NB = 5 THEN 5000
510  IF NB = 6 THEN 6000
520  IF NB = 7 THEN 7000
```

```
530  IF NB = 8 THEN 8000
540 :
550 :
1000  REM     **--SINGLE POINT--**
1010  PRINT D$;"RUN LOW.HIGH.POINT"
1980 :
1990 :
2000  REM    **--HORIZONTAL LINE--**
2010  PRINT D$;"RUN LOW.HIGH.HORIZ"
2980 :
2990 :
3000  REM    **--VERTICAL LINE--**
3010  PRINT D$;"RUN LOW.HIGH.VERT"
3980 :
3990 :
4000  REM   **--COLORS--**
4010  PRINT D$;"RUN LOW.HIGH.COLOR"
4980 :
4990 :
5000  REM   **--CIRCLE--**
5010  PRINT D$;"RUN LOW.HIGH.CIRCLE"
5980 :
5990 :
6000  REM   **--RECTANGLE--**
6010  PRINT D$;"RUN LOW.HIGH.RECT"
6980 :
6990 :
7000  REM     **--DISPLAY CATALOG--**
7010  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
7020  HOME
7030  PRINT D$;"CATALOG"
7040  PRINT
7050  POKE  - 16368,0: REM   CLEAR KBRD BUFFER
7060  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT " K
    EY TO CONTINUE:";L$
7070  GOTO 170
7980 :
7990 :
8000  REM   **--END--**
8010  HOME
8020  VTAB 10
8030  INVERSE
8040  PRINT "SEE YOU NEXT TIME."
8050  NORMAL
8060  END
9998 :
9999 :
10000  REM  LOW/HIGH COMPARISON
```

```
11000  REM  COPYRIGHT OCTOBER 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS RD
14000  REM  CORVALLIS OR. 97330-9340
```

Low- and High-Resolution Point Display

```
100  REM  ***--LOW.HIGH.POINT--***
110 :
120 :
130  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
150  HOME : VTAB 3
160  HTAB 12: INVERSE : PRINT "SINGLE POINT": NORMAL
170  PRINT : PRINT : PRINT
180  HTAB 11: PRINT "LOW RESOLUTION": PRINT : PRINT : HTAB 17: INVERSE :
     PRINT "VS": NORMAL
190  PRINT : PRINT
200  HTAB 10: PRINT "HIGH RESOLUTION"
210  GOSUB 8000: REM  RETURN KEY ROUTINE
220  HOME : VTAB 3: INVERSE : HTAB 14: PRINT "INSTRUCTIONS": NORMAL
230  VTAB 10: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY FOR COMPARISONS."
240  PRINT : PRINT
250  PRINT : PRINT "PRESS THE '";: INVERSE : PRINT "S";: NORMAL : PRINT
    "' KEY TO STOP COMPARISON."
260  PRINT : PRINT
270  PRINT : PRINT "PRESS ";: INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT
    " TO BEGIN COMPARISON.";
280  GET L$
290  IF L$ = "S" OR L$ = "s" THEN 5000
300  PRINT
310  IF  ASC (L$) <  > 32 THEN 230
320  HGR : GR
330  COLOR= 9
340  PLOT 20,20
350  HOME : VTAB 23
360  GOSUB 1000: REM  LOW-RES
370  IF L$ = "S" OR L$ = "s" THEN 5000
380  POKE  - 16297,0: REM  SWITCH DISPLAY TO HI-RES
390  HCOLOR= 6
400  HPLOT 142,75
410  HCOLOR= 5
420  HPLOT 139,80 TO 145,80
430  HPLOT 139,81 TO 145,81
440  HPLOT 139,82 TO 145,82
```

```
450  HPLOT 139,83 TO 145,83
460  VTAB 23: CALL  - 868: REM   CLEAR LINE
470  GOSUB 2000: REM  HI-RES
480  POKE  - 16298,0: REM  SWITCH DISPLAY TO LOW-RES
490  VTAB 23: CALL  - 868: REM   CLEAR LINE
500  GOSUB 1000: REM  LOW-RES
510  IF L$ = "S" OR L$ = "s" THEN 5000
520  POKE  - 16297,0: REM  SWITCH DISPLAY TO HI-RES
530  VTAB 23: CALL  - 868: REM   CLEAR LINE
540  GOSUB 2000: REM  HI-RES
550  IF L$ = "S" OR L$ ="s" THEN 5000
560  GOTO 480: REM  CONTINUE COMPARISON
570 :
580 :
1000  REM  **--LOW RESOLUTION--**
1010  INVERSE : PRINT "LOW";: NORMAL : PRINT " RESOLUTION--ONE POINT:";:
     GET L$
1020  PRINT
1030  RETURN
1980 :
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  INVERSE : PRINT "HIGH";: NORMAL : PRINT " RES.--1 PT.& 7 X 4 RECT.
    (28 PTS.):";: GET L$
2020  PRINT
2030  RETURN
2980 :
2990 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 5
5020  PRINT "It takes approximately 28 high-"
5030  PRINT
5040  PRINT "resolution points to equal the size of"
5050  PRINT
5060  PRINT "one low-resolution point, or each high-"
5070  PRINT
5080  PRINT "resolution point is approximately 1/28"
5090  PRINT
5100  PRINT "the size of a low-resolution point."
5110  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
5120  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU. ";L$
5130  PRINT D$;"RUN LOW.HIGH.MENU"
5990 :
5995 :
```

```
8000  REM  **--RETURN KEY ROUTINE--**
8010  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY TO CONTINUE: ";L$
8020  RETURN
9998 :
9999 :
10000  REM  LOW/HIGH COMPARISON
11000  REM  COPYRIGHT OCTOBER 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS RD
14000  REM  CORVALLIS OR. 97330-9340
```

Low- and High-Resolution Horizontal-Line Display

```
100  REM  ***--LOW.HIGH.HORIZ--***
110 :
120 :
130  POKE  - 16368,0: REM   CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
150  HOME : VTAB 3
160  HTAB 10: INVERSE : PRINT "HORIZONTAL LINE": NORMAL
170  PRINT : PRINT : PRINT
180  HTAB 11: PRINT "LOW RESOLUTION": PRINT : PRINT : HTAB 17: INVERSE :
     PRINT "VS": NORMAL
190  PRINT : PRINT
200  HTAB 10: PRINT "HIGH RESOLUTION"
210  GOSUB 8000: REM   RETURN KEY ROUTINE
220  HOME : VTAB 3: INVERSE : HTAB 14: PRINT "INSTRUCTIONS": NORMAL
230  VTAB 10: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY FOR COMPARISONS."
240  PRINT : PRINT
250  PRINT : PRINT "PRESS THE '";: INVERSE : PRINT "S";: NORMAL : PRINT
    "' KEY TO STOP COMPARISON."
260  PRINT : PRINT
270  PRINT : PRINT "PRESS ";: INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT
    " TO BEGIN COMPARISON.";
280  GET L$
290  IF L$ = "S" OR L$ = "s" THEN 5000
300  PRINT
310  IF  ASC (L$) <  > 32 THEN 230
320  HGR : GR : COLOR= 1
330  HLIN 0,39 AT 20
340  HOME : VTAB 23
350  GOSUB 1000: REM   LOW-RES
```

```
360  IF L$ = "S" OR L$ = "s" THEN 5000
370  POKE  - 16297,0: REM   SWITCH DISPLAY TO HI-RES
380  HCOLOR= 6
390  HPLOT 0,80 TO 279,80
400  VTAB 23: CALL  - 868: REM  CLEAR LINE
410  GOSUB 2000: REM   HI-RES
420  POKE  - 16298,0: REM   SWITCH DISPLAY TO LOW-RES
430  VTAB 23: CALL  - 868: REM  CLEAR LINE
440  GOSUB 1000: REM   LOW-RES
450  IF L$ = "S" OR L$ = "s" THEN 5000
460  POKE  - 16297,0: REM   SWITCH DISPLAY TO HI-RES
470  VTAB 23: CALL  - 868: REM  CLEAR LINE
480  GOSUB 2000: REM   HI-RES
490  IF L$ = "S" OR L$ = "s" THEN 5000
500  GOTO 420: REM  CONTINUE COMPARISON
510 :
520 :
1000  REM  **--LOW RESOLUTION--**
1010  INVERSE : PRINT "LOW";: NORMAL : PRINT " RES.--HORIZ. LINE (40 POI
    NTS):";: GET L$
1020  PRINT
1030  RETURN
1980 :
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  INVERSE : PRINT "HIGH";: NORMAL; : PRINT " RES.--HORIZ. LINE (280 P
    OINTS):";: GET L$
2020  PRINT
2030  RETURN
2980 :
2990 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 5
5020  PRINT "--Each low-resolution horizontal line"
5030  PRINT
5040  PRINT "  can have up to 40 points numbered 0"
5050  PRINT
5060  PRINT "  to 39."
5070  PRINT
5080  PRINT "--Each high-resolution horizontal Line"
5090  PRINT
5100  PRINT "  can have up to 280 points numbered"
5110  POKE  - 16368,0: REM   CLEAR KBRD BUFFER
5120  PRINT
5130  PRINT "  0 to 279."
5140  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU. ";L$
5150  PRINT D$;"RUN LOW.HIGH.MENU"
```

```
5160 :
5170 :
8000  REM  **--RETURN KEY ROUTINE--**
8010  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY TO CONTINUE: ";L$
8020  RETURN
9998 :
9999 :
10000  REM  LOW/HIGH COMPARISON
11000  REM  COPYRIGHT OCTOBER 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS RD
14000  REM  CORVALLIS OR. 97330-9340
```

Low- and High-Resolution Vertical-Line Display

```
100  REM  ***--LOW.HIGH.VERT--***
110 :
120 :
130  POKE  - 16368,0: REM   CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
150  HOME : VTAB 3
160  HTAB 12: INVERSE : PRINT "VERTICAL LINE": NORMAL
170  PRINT : PRINT : PRINT
180  HTAB 11: PRINT "LOW RESOLUTION": PRINT : PRINT : HTAB 17: INVERSE :
     PRINT "VS": NORMAL
190  PRINT : PRINT
200  HTAB 10: PRINT "HIGH RESOLUTION"
210  GOSUB 8000: REM   RETURN KEY ROUTINE
220  HOME : VTAB 3: INVERSE : HTAB 14: PRINT "INSTRUCTIONS": NORMAL
230  VTAB 10: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY FOR COMPARISONS."
240  PRINT : PRINT
250  PRINT : PRINT "PRESS THE '";: INVERSE : PRINT "S";: NORMAL : PRINT
    "' KEY TO STOP COMPARISON."
260  PRINT : PRINT
270  PRINT : PRINT "PRESS ";: INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT
    " TO BEGIN COMPARISON.";
280  GET L$
290  IF L$ = "S" OR L$ = "s" THEN 5000
300  PRINT
310  IF  ASC (L$) <  > 32 THEN 230
320  HGR : GR
330  COLOR= 13
340  VLIN 0,39 AT 20
350  HOME : VTAB 23
360  GOSUB 1000: REM  LOW-RES INFO
```

```
370  IF L$ = "S" OR L$ = "s" THEN 5000
380  POKE  - 16297,0: REM  SWITCH DISPLAY TO HI-RES
390  HCOLOR= 1
400  HPLOT 139,0 TO 139,159
410  VTAB 23: CALL  - 868: REM  CLEAR LINE
420  GOSUB 2000: REM  HI-RES INFO
430  POKE  - 16298,0: REM  SWITCH DISPLAY TO LOW-RES
440  VTAB 23: CALL  - 868: REM  CLEAR LINE
450  GOSUB 1000: REM  LOW-RES INFO
460  IF L$ = "S" OR L$ = "s" THEN 5000
470  POKE  - 16297,0: REM  SWITCH DISPLAY TO HI-RES
480  VTAB 23: CALL  - 868: REM  CLEAR LINE
490  GOSUB 2000: REM  HI-RES INFO
500  IF L$ = "S" OR L$ = "s" THEN 5000
510  GOTO 430: REM CONTINUE COMPARISON
520 :
530 :
1000  REM  **--LOW RESOLUTION--**
1010  INVERSE : PRINT "LOW";: NORMAL : PRINT " RES.--VERT. LINE (40 POIN
    TS):";: GET L$
1020  PRINT
1030  RETURN
1980 :
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  INVERSE : PRINT "HIGH";: NORMAL : PRINT " RES.--VERT. LINE (160 PO
    INTS):";: GET L$
2020  PRINT
2030  RETURN
2980 :
2990 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 5
5020  PRINT "--Each low-resolution vertical line"
5030  PRINT
5040  PRINT "  can have up to 40 points numbered 0"
5050  PRINT
5060  PRINT "  to 39."
5070  PRINT
5080  PRINT "--Each high-resolution vertical line"
5090  PRINT
5100  PRINT "  can have up to 160 points numbered"
5110  POKE  - 16368,0: REM   CLEAR KBRD BUFFER
5120  PRINT
5130  PRINT "  0 to 159."
5140  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU. ";L$
5150  PRINT D$;"RUN LOW.HIGH.MENU"
```

```
5160 :
5170 :
8000  REM  **--RETURN KEY ROUTINE--**
8010  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY TO CONTINUE: ";L$
8020  RETURN
9998 :
9999 :
10000  REM  LOW/HIGH COMPARISON
11000  REM  COPYRIGHT OCTOBER 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS RD
14000  REM  CORVALLIS OR. 97330-9340
```

Low- and High-Resolution Color Display

```
100  REM  ***--LOW.HIGH.COLOR--***
110 :
120 :
130  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
150  HOME : VTAB 3
160  HTAB 15: INVERSE : PRINT "COLORS": NORMAL
170  PRINT : PRINT : PRINT
180  HTAB 11: PRINT "LOW RESOLUTION": PRINT : PRINT : HTAB 17: INVERSE :
     PRINT "VS": NORMAL
190  PRINT : PRINT
200  HTAB 10: PRINT "HIGH RESOLUTION"
210  GOSUB 8000: REM  RETURN KEY ROUTINE
220  HOME : VTAB 3: INVERSE : HTAB 14: PRINT "INSTRUCTIONS": NORMAL
230  VTAB 10: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY FOR COMPARISONS."
240  PRINT : PRINT
250  PRINT : PRINT "PRESS THE '";: INVERSE : PRINT "S";: NORMAL : PRINT
    "' KEY TO STOP COMPARISON."
260  PRINT : PRINT
270  PRINT : PRINT "PRESS ";: INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT
    " TO BEGIN COMPARISON.";
280  GET L$
290  IF L$ = "S" OR L$ = "s" THEN 5000
300  PRINT
310  IF  ASC (L$) <  > 32 THEN 230
320  HOME : HGR : GR
330  FOR I = 2 TO 32 STEP 2
340  COLOR= C
350  VLIN 0,39 AT I
360 C = C + 1
```

```
370  NEXT I
380  HOME : VTAB 21
390  GOSUB 1000: REM LOW-RES INFO
400  IF L$ = "S" OR L$ = "s" THEN 5000
410  POKE  - 16297,0: REM  SWITCH DISPLAY TO HI-RES
420  HCOLOR= 0: HPLOT 50,0 TO 50,159
430  HCOLOR= 1: HPLOT 73,0 TO 73,159
440  HCOLOR= 2: HPLOT 94,0 TO 94,159
450  HCOLOR= 3: HPLOT 115,0 TO 115,159
460  HCOLOR= 3: HPLOT 116,0 TO 116,159
470  HCOLOR= 4: HPLOT 132,0 TO 132,159
480  HCOLOR= 5: HPLOT 155,0 TO 155,159
490  HCOLOR= 6: HPLOT 178,0 TO 178,159
500  HCOLOR= 7: HPLOT 197,0 TO 197,159
510  HCOLOR= 7: HPLOT 198,0 TO 198,159
520  HOME
530  VTAB 21: CALL  - 868: REM  CLEAR LINE
540  GOSUB 2000: REM  HI-RES INFO
550  POKE  - 16298,0: REM  SWITCH DISPLAY TO LOW-RES
560  HOME
570  VTAB 21: CALL  - 868: REM  CLEAR LINE
580  GOSUB 1000: REM LOW-RES INFO
590  IF L$ = "S" OR L$ = "s" THEN 5000
600  POKE  - 16297,0: REM  SWITCH TO HI-RES
610  HOME
620  VTAB 21: CALL  - 868: REM  CLEAR LINE
630  GOSUB 2000: REM  HI-RES INFO
640  IF L$ = "S" OR L$ = "s" THEN 5000
650  GOTO 550: REM  CONTINUE COMPARISON
660 :
670 :
1000  REM  **--LOW RESOLUTION--**
1010  HTAB 3: PRINT "0 1 2 3 4 5 6 7 8 9 1 1 1 1 1 1"
1020  HTAB 24: PRINT "0 1 2 3 4 5"
1030  PRINT
1040  HTAB 3
1050  INVERSE : PRINT "LOW";: NORMAL : PRINT " RES.--16 COLORS (0 TO 15)
    :";: GET L$
1060  PRINT
1070  RETURN
1980 :
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  HTAB 8: PRINT "0  1  2  3  4  5  6  7"
2020  PRINT : PRINT
2030  INVERSE : PRINT "HIGH";: NORMAL : PRINT " RES.--6 COLORS (0 TO 7,
    2B, 2W):";: GET L$
2040  PRINT
```

```
2050  RETURN
2980 :
2990 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 4
5020  PRINT "--Low-resolution has 16 color"
5030  PRINT
5040  PRINT "  capability with colors numbered"
5050  PRINT
5060  PRINT "  0 to 15."
5070  PRINT
5080  PRINT "--High-resolution has 6 color"
5090  PRINT
5100  PRINT "  capability with colors numbered"
5110  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
5120  PRINT
5130  PRINT "  0 to 7 (#'s 0 & 4 are BLACK,"
5140  PRINT
5150  PRINT "  3 and 7 are WHITE)."
5160  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU. ";L$
5170  PRINT D$;"RUN LOW.HIGH.MENU"
5180 :
5190 :
8000  REM  **--RETURN KEY ROUTINE--**
8010  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY TO CONTINUE: ";L$
8020  RETURN
9998 :
9999 :
10000  REM  LOW/HIGH COMPARISON
11000  REM  COPYRIGHT OCTOBER 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS RD
14000  REM  CORVALLIS OR. 97330-9340
```

Low- and High-Resolution Circle Display

```
100  REM  ***--LOW.HIGH.CIRCLE--***
110 :
120 :
130  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
150  HOME : VTAB 3
160  HTAB 14: INVERSE : PRINT "CIRCLES": NORMAL
170  PRINT : PRINT : PRINT
180  HTAB 11: PRINT "LOW RESOLUTION": PRINT : PRINT : HTAB 17: INVERSE :
```

```
    PRINT "VS": NORMAL
190 PRINT : PRINT
200 HTAB 10: PRINT "HIGH RESOLUTION"
210 GOSUB 8000: REM  RETURN KEY ROUTINE
220 HOME : VTAB 3: INVERSE : HTAB 14: PRINT "INSTRUCTIONS": NORMAL
230 VTAB 10: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
   " KEY FOR COMPARISONS."
240 PRINT : PRINT
250 PRINT : PRINT "PRESS THE '";: INVERSE : PRINT "S";: NORMAL : PRINT
   "' KEY TO STOP COMPARISON."
260 PRINT : PRINT
270 PRINT : PRINT "PRESS ";: INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT
   " TO BEGIN COMPARISON.";
280 GET L$
290 IF L$ = "S" OR L$ = "s" THEN 5000
300 PRINT
310 IF  ASC (L$) < > 32 THEN 230
320 HOME
330 HGR : GR
340 COLOR= 13
350 X1 = 20:Y1 = 20:P = .75:R = 11
360 FOR K = 0 TO 6.4 STEP .05
370 GOSUB 3000: REM  CIRCLE ROUTINE
380 PLOT X,Y
390 NEXT K
400 VTAB 23
410 GOSUB 1000: REM LOW-RES INFO
420 IF L$ = "S" OR L$ = "s" THEN 5000
430 HOME
440 POKE  - 16297,0: REM  SWITCH DISPLAY TO HI-RES
450 HCOLOR= 3
460 X1 = 140:Y1 = 80:P = 1.16:R = 45
470 FOR K = 0 TO 6.4 STEP .05
480 GOSUB 3000: REM  CIRCLE ROUTINE
490 IF K = 0 THEN  HPLOT X,Y: GOTO 510
500 HPLOT  TO X,Y
510 NEXT K
520 VTAB 23: CALL  - 868: REM  CLEAR LINE
530 GOSUB 2000: REM  HI-RES INFO
540 POKE  - 16298,0: REM  SWITCH DISPLAY TO LOW-RES
550 VTAB 23: CALL  - 868: REM  CLEAR LINE
560 GOSUB 1000: REM LOW-RES INFO
570 IF L$ = "S" OR L$ = "s" THEN 5000
580 POKE  - 16297,0: REM  SWITCH DISPLAY TO HI-RES
590 VTAB 23: CALL  - 868: REM  CLEAR LINE
600 GOSUB 2000: REM  HI-RES INFO
610 IF L$ = "S" OR L$ = "s" THEN 5000
620 GOTO 540: REM  CONTINUE COMPARISON
```

```
630 :
640 :
1000  REM  **--LOW RESOLUTION--**
1010  HTAB 10: INVERSE : PRINT "LOW";: NORMAL : PRINT " RESOLUTION--CIRC
    LE:";: GET L$
1020  PRINT
1030  RETURN
1980 :
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  HTAB 10: INVERSE : PRINT "HIGH";: NORMAL : PRINT " RESOLUTION--CIR
    CLE:";: GET L$
2020  PRINT
2030  RETURN
2980 :
2990 :
3000  REM  **--CIRCLE ROUTINE--**
3010 X2 = R *  COS (K)
3020 Y2 = R *  SIN (K)
3030 X = P * X2 + X1
3040 Y = Y1 - Y2
3050  RETURN
3980 :
3990 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 5
5020  PRINT "--Both the low-resolution and high-"
5030  PRINT
5040  PRINT "resolution circles were drawn with"
5050  PRINT
5060  PRINT "virtually the same routine. This"
5070  PRINT
5080  PRINT "demonstration provides a good"
5090  PRINT
5100  PRINT "example of the reason most serious"
5110  PRINT
5120  PRINT "graphic work (including games)"
5130  PRINT
5140  PRINT "is done in high-resolution."
5150  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
5160  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU. ";L$
5170  PRINT D$;"RUN LOW.HIGH.MENU"
5180 :
5190 :
8000  REM  **--RETURN KEY ROUTINE--**
8010  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY TO CONTINUE: ";L$
```

```
8020  RETURN
9998 :
9999 :
10000  REM  LOW/HIGH COMPARISON
11000  REM  COPYRIGHT OCTOBER 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS RD
14000  REM  CORVALLIS OR. 97330-9340
```

Low- and High-Resolution Rectangle Display

```
100  REM  ***--LOW.HIGH.RECT--***
110 :
120 :
130  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
150  HOME : VTAB 3
160  HTAB 13: INVERSE : PRINT "RECTANGLES": NORMAL
170  PRINT : PRINT : PRINT
180  HTAB 11: PRINT "LOW RESOLUTION": PRINT : PRINT : HTAB 17: INVERSE :
     PRINT "VS": NORMAL
190  PRINT : PRINT
200  HTAB 10: PRINT "HIGH RESOLUTION"
210  GOSUB 8000: REM  RETURN KEY ROUTINE
220  HOME : VTAB 3: INVERSE : HTAB 14: PRINT "INSTRUCTIONS": NORMAL
230  VTAB 10: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY FOR COMPARISONS."
240  PRINT : PRINT
250  PRINT : PRINT "PRESS THE '";: INVERSE : PRINT "S";: NORMAL : PRINT
    "' KEY TO STOP COMPARISON."
260  PRINT : PRINT
270  PRINT : PRINT "PRESS ";: INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT
    " TO BEGIN COMPARISON.";
280  GET L$
290  IF L$ = "S" OR L$ = "s" THEN 5000
300  PRINT
310  IF  ASC (L$) <  > 32 THEN 230
320  HOME
330  HGR : GR
340  COLOR= 13
350 X = 20:Y = 20:N = 1
360  PLOT X,Y - 1
370  GOSUB 3000: REM  CIRCLE ROUTINE
380  VTAB 23
390  GOSUB 1000: REM  LOW-RES INFO
400  IF L$ = "S" OR L$ = "s" THEN 5000
410  HOME
```

```
420  POKE  - 16297,0: REM  SWITCH DISPLAY TO HI-RES
430  HCOLOR= 3
440 X = 140:Y = 80:N = 1
450  HPLOT X,Y - 1
460  GOSUB 4000: REM  CIRCLE ROUTINE
470  VTAB 23: CALL  - 868: REM  CLEAR LINE
480  GOSUB 2000: REM  HI-RES INFO
490  POKE  - 16298,0: REM  SWITCH DISPLAY TO LOW-RES
500  VTAB 23: CALL  - 868: REM  CLEAR LINE
510  GOSUB 1000: REM LOW-RES INFO
520  IF L$ = "S" OR L$ = "s" THEN 5000
530  POKE  - 16297,0: REM  SWITCH DISPLAY TO HI-RES
540  VTAB 23: CALL  - 868: REM  CLEAR LINE
550  GOSUB 2000: REM  HI-RES INFO
560  IF L$ = "S" OR L$ = "s" THEN 5000
570  GOTO 490: REM  CONTINUE COMPARISON
580 :
590 :
1000  REM  **--LOW RESOLUTION--**
1010  INVERSE : PRINT "LOW";: NORMAL : PRINT " RESOLUTION--RECTANGLE WIT
   HIN RECT:";: GET L$
1020  PRINT
1030  RETURN
1980 :
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  INVERSE : PRINT "HIGH";: NORMAL : PRINT " RESOLUTION--RECTANGLE WI
   THIN RECT:";: GET L$
2020  PRINT
2030  RETURN
2980 :
2990 :
3000  REM   **--LOW RES.RECTANGLE ROUTINE--**
3010  COLOR=  INT (16 *  RND (1))
3020  FOR J = 0 TO N
3030  PLOT X + J,Y
3040  NEXT J
3050 X = X + N
3060 N = N + 1
3070  FOR J = 0 TO N
3080  PLOT X,Y - J
3090  NEXT J
3100 Y = Y - N
3110  FOR J = 0 TO N
3120  PLOT X - J,Y
3130  NEXT J
3140 X = X - N
3150  FOR J = 0 TO N
```

```
3160  PLOT X,Y + J
3170  NEXT J
3180 N = N + 1
3190 Y = Y + N
3200  IF N < 38 THEN 3000
3210  RETURN
3980 :
3990 :
4000  REM    **--HIGH RES.RECTANGLE ROUTINE--**
4010  HCOLOR=  INT (8 *  RND (1))
4020  FOR J = 0 TO N
4030  HPLOT X + J,Y
4040  NEXT J
4050 X = X + N
4060 N = N + 1
4070  FOR J = 0 TO N
4080  HPLOT X,Y - J
4090  NEXT J
4100 Y = Y - N
4110  FOR J = 0 TO N
4120  HPLOT X - J,Y
4130  NEXT J
4140 X = X - N
4150  FOR J = 0 TO N
4160  HPLOT X,Y + J
4170  NEXT J
4180 N = N + 1
4190 Y = Y + N
4200  IF N < 158 THEN 4000
4210  RETURN
4980 :
4990 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 2
5020  PRINT "Did you notice that high-resolution"
5030  PRINT
5040  PRINT "lines changed colors, disappeared,"
5050  PRINT
5060  PRINT "or expanded? This demonstration"
5070  PRINT
5080  PRINT "provides a good example of some of the"
5090  PRINT
5100  PRINT "problems associated with color in"
5110  PRINT
5120  PRINT "the high-resolution mode. Even"
5130  PRINT
5140  PRINT "numbered columns will display even"
5150  PRINT
```

```
5160  PRINT "numbered colors. Odd numbered columns"
5170  PRINT
5180  PRINT "will display odd numbered colors."
5190  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
5200  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU. ";L$
5210  PRINT D$;"RUN LOW.HIGH.MENU"
5220 :
5230 :
8000  REM  **--RETURN KEY ROUTINE--**
8010  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY TO CONTINUE: ";L$
8020  RETURN
9998 :
9999 :
10000  REM  LOW/HIGH COMPARISON
11000  REM  COPYRIGHT OCTOBER 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS RD
14000  REM  CORVALLIS OR. 97330-9340
```

CHAPTER 6

Low-Resolution Graphics

As we have already seen, the usual method of beginning low-resolution graphics is with the GR (for GRaphics) command. Once low-resolution graphics are established, the color must be set. At this point, you have a choice of commands:

1. The PLOT command
2. THE HLIN (horizontal line) command
3. The VLIN (vertical line) command
4. The SCRN (color of the screen at a specified location) command

SCREEN-DISPLAY TUTORIAL

We will begin with a series of programs that make use of large letters created in low-resolution graphics. As designed, these letters will appear in the same location on the screen every time they are used. They will also be the same size every time. A more versatile routine would allow each character to be placed anywhere on the screen and to be a different size depending upon the specific application. However, the complexity of an all-purpose letter-graphing routine is beyond the scope of this book. The MOVING.I program at the end of this chapter, however, gives some techniques that can be used to create different size letters in different positions.

Each of the letters will be formed using a combination of VLIN, HLIN and PLOT instructions. The first decision involves the vertical size of the characters. Arbitrarily, I chose to have the letters extend over 30 rows, beginning at the 5th row and ending at the 35th row. Figure 6.1 illustrates the boundaries for low-resolution letters. Some letters, for appearance's sake, extend only to the 34th row, while others go beyond the 35th row.

The horizontal size of the letters varies and depends on the character being drawn. A common center point was chosen, so that the width of the characters fell equally on each side

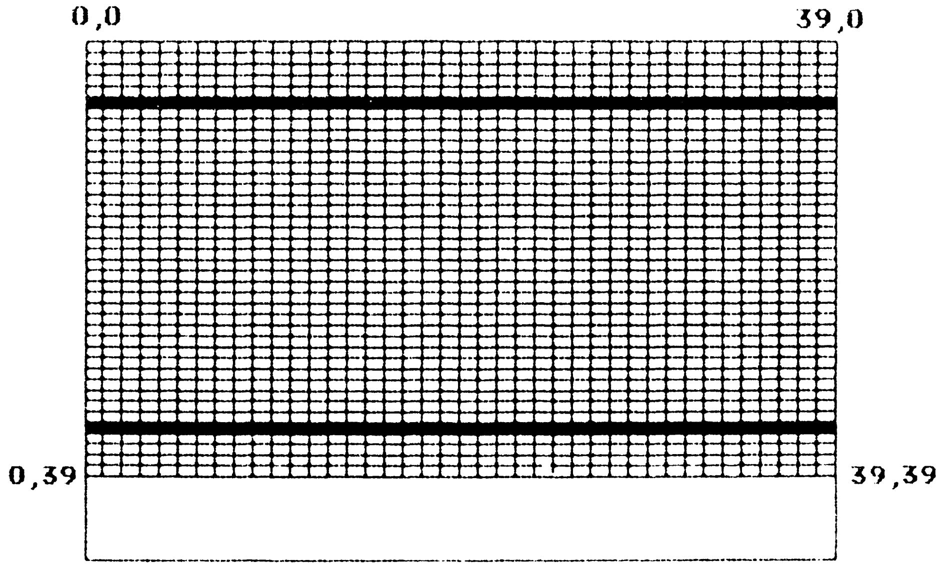


FIGURE 6.1. Horizontal boundaries for low-resolution letters program.

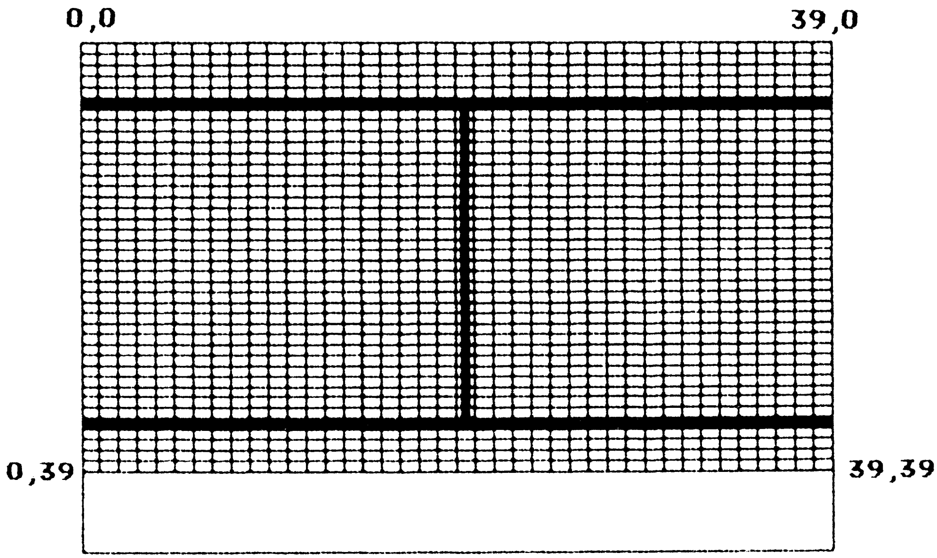


FIGURE 6.2. Vertical axis for low-resolution letters program.

of the center point. Since there are 40 horizontal points or columns, the 20th column was chosen as the axis around which characters would be formed. With these specifications in mind, the characters can be drawn within this grid using the instructions available for the low-resolution screen. Figure 6.2 illustrates the vertical axis used in the letters program.

Characters are formed on the screen in much the same way they are formed using a pen or pencil. For example, the uppercase character "I" is drawn with a vertical line, a shorter, horizontal line at the top and another horizontal line of the same length at the bottom. The same procedure is used to form an uppercase "I" on the computer's low-resolution screen. First, instruct the computer to draw a vertical line from the 5th row to the 35th row on our "axis" or horizontal center line (column 20). The computer instruction that accomplishes this is

```
VLIN 5,35 AT 20
```

(Remember, the graphic mode must first be set with the GR command and the color established: COLOR = 6.)

The VLIN instruction tells the computer to draw a vertical line beginning at the 5th row and extending to the 35th row (5,35) on the 20th column (AT 20) (Fig. 6.3).

The same logic is used to draw the two horizontal lines that form the top and bottom of the uppercase "I". Since the vertical line is 30 points in length, an appropriate size for the horizontal lines might be one third of that size, or 10 points in length. Using the axis of 20, and the length of 10 (actually 11 points when the axis point itself is counted), the beginning point of the horizontal line is determined by subtracting half the number of points in the line from the axis point itself: 10 ÷ 2 = 5; 20 − 5 = 15. In other words, half the horizontal line goes before the axis point and the other half goes after the axis point. Therefore, both horizontal lines (the top and bottom of the "I") must start at the 15th column and end at the 25th column. Since we have already determined that the tops of the characters will be on the 5th row, the instruction to draw the top horizontal line would be

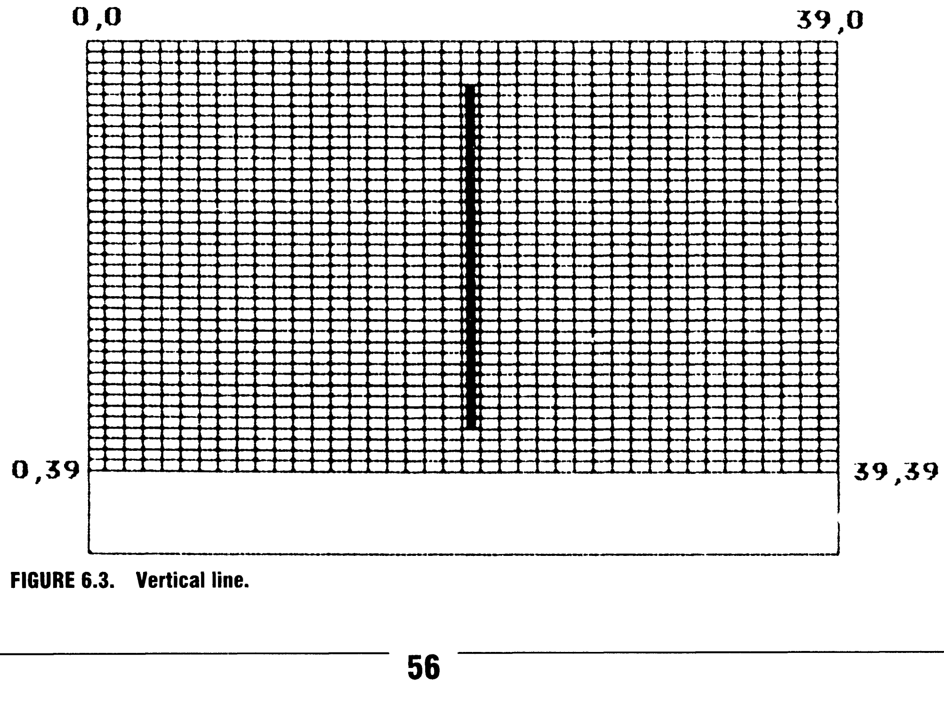


FIGURE 6.3. Vertical line.

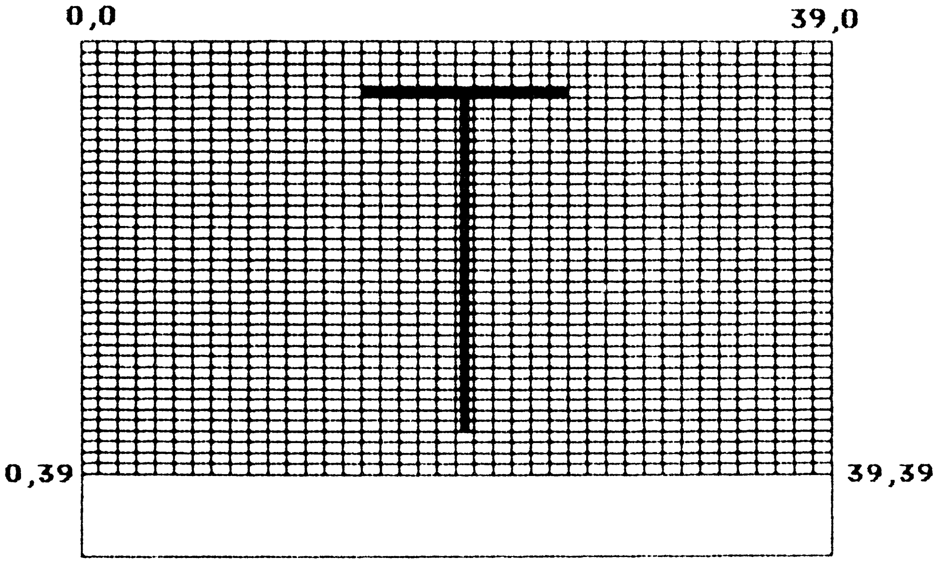


FIGURE 6.4. Top horizontal line.

```
HLIN 15,25 AT 5
```

The HLIN instruction tells the computer to draw a horizontal line from the 15th column to the 25th column (15,25) on the 5th row (AT 5) (Fig. 6.4).

The bottom line of the uppercase "I" will also go from the 15th to the 25th column, but since the bottoms of the characters are to be on the 35th row, the instruction to draw the bottom horizontal line is

```
HLIN 15,25 AT 35
```

This instruction tells the computer to draw a horizontal line (HLIN) from the 15th column to the 25th column (15,25) on the 35th row (AT 35) (Fig. 6.5).

This HLIN instruction completes the uppercase "I."

Screen Dimensions

Once the concept of using screen dimensions to form characters on the low-resolution screen is learned, most of the struggle to understand low-resolution graphics is over. The hardest part for most people is understanding and thinking in terms of a 40-column by 40-row screen whose dimensions begin in the upper left-hand corner with point 0,0 and proceed down the screen to the lower right-hand corner. To many, especially those with some math, these dimensions are not what they should be. For reasons that originally

related to the hardware (i.e., the chips and the way monitors and televisions receive their signals), manufacturers almost universally installed this form of graphics dimensioning within their computers. Just remember that you count down the screen from row 0 to row 39 (47 for full-screen graphics) and that you count across the screen from left to right beginning with column 0 and ending with column 39.

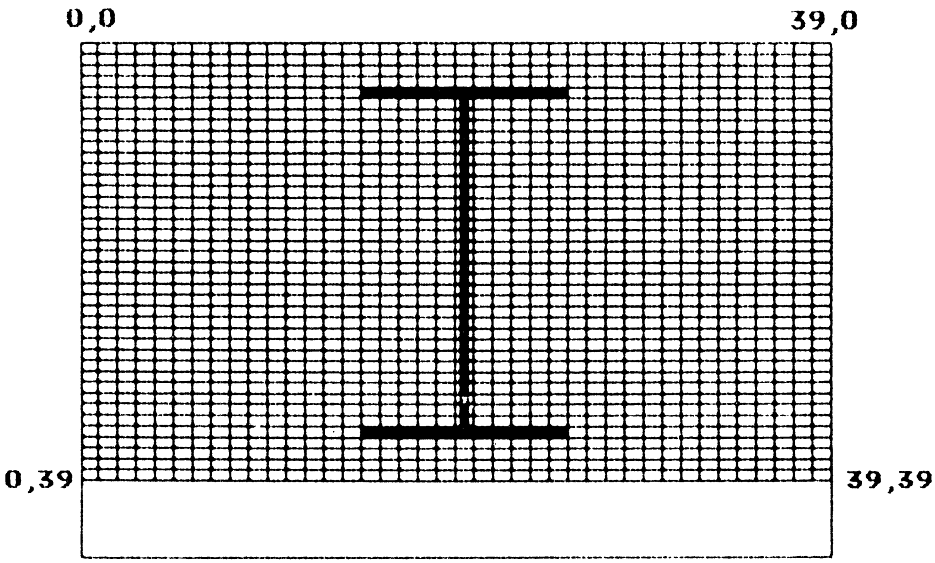


FIGURE 6.5. Bottom horizontal line.

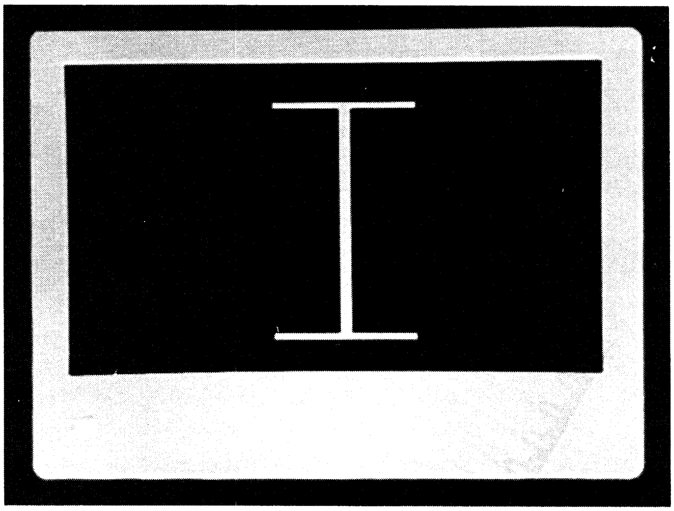


FIGURE 6.6. Low-resolution capital letter I.

The full set of instructions to form a blue capital letter "I" (Fig 6.6) are

```
GR
COLOR = 6
VLIN 5,35 AT 20
HLIN 15,25 AT 5
HLIN 15,25 AT 35
```

Numbering Your Program

To put these instructions into the form of a program, you need only add a line number before each instruction. The choice of line numbers is up to you, but space should be left between each number to allow for the possibility of additional instructions. Thus the program listing would be

```
100  GR
110  COLOR = 6
120  VLIN 5,35 AT 20
130  HLIN 15,25 AT 5
140  HLIN 15,25 AT 35
150  END
```

Although these are the only instructions necessary to create a blue capital "I" in the middle of the screen, a few additional lines will make it easier for you or for another programmer to read the listing and understand exactly what the program is supposed to do. BASIC provides a word, REM (for REMark), that allows you to add comments about the program without affecting the program execution. A line containing a REM statement will execute properly, but anything following the reserved word "REM" is treated as a remark and is ignored by the computer. The judicious use of REMarks can make a program very easy to understand.

To help set off the parts of a program, a colon can be used in two ways to add to a program's readability. The colon is used to separate instructions that appear together on the same line number. For example, line 100 could be

```
100 GR:COLOR = 6
```

Line 100 would then contain two instructions separated by a colon that the computer would process one after the other. A colon can also be used on a line by itself to simply hold the line number in place. If I want to separate the instructions on lines 120, 130, and 140 from the rest of the program, I can use colons to hold other line numbers in place before and after these line numbers.

```
100  GR
110  COLOR = 6
115  :
117  :
120  VLIN 5,35 AT 20
130  HLIN 15,25 AT 5
140  HLIN 15,25 AT 35
145  :
147  :
150  END
```

Now I am able to clearly see the three parts of this program. The first part sets up or initializes the computer for what the programmer wants to do (use the low-resolution screen). The second part draws the letter "I," and the third part concludes the program. If we add these or similar comments to this program using the REM statement, the program is more readable:

```
90  REM **--INITIALIZE LOW-RES GRAPHICS--**
100 GR
110 COLOR = 6: REM SET COLOR TO BLUE
115 :
117 :
119 REM **--DRAW CAPITAL LETTER I--**
120 VLIN 5,35 AT 20
130 HLIN 15,25 AT 5
140 HLIN 15,25 AT 35
145 :
147 :
149 REM **--END PROGRAM--**
150 END
```

I have found it very beneficial to use REM to include the actual name of the program, as it appears in the diskette's CATALOG, as the first line in my programs. Since the program name is chosen by the programmer, you can name your program anything you want, but it is best to choose a name that helps clarify the intention of the program. A name such as "LOW.RES.I" gives enough information that anyone cataloguing a diskette or listing the program will have a fair idea what the program does. The final version is

```
50  REM ***--LOW.RES.I--***
60  :
70  :
90   REM **--INITIALIZE LOW-RES GRAPHICS--**
100  GR
110 COLOR = 6: REM SET COLOR TO BLUE
115 :
117 :
```

```
119 REM  **--DRAW CAPITAL LETTER I--**
120 VLIN 5,35 AT 20
130 HLIN 15,25 AT 5
140 HLIN 15,25 AT 35
145 :
147 :
149 REM **--END PROGRAM--**
150 END
```

One last note. I have added the *'s and -'s on the REM line because I think it helps to set off the actual remark. In addition, I purposely left out line number 80 to show that line numbers do not need to have specific sequence or pattern.

I prefer to begin each program with the program name on line 100, the first line in the program. Further, I like to have each part or section of the program begin on a round number (100, 500, 1000, 2000, etc.) and each line number to be a multiple of ten. This numbering system is not always possible and, if you don't have a renumbering program, is sometimes more trouble that it is worth. But if you have access to a utility program that can renumber your program (such as the RENUMBER program on the DOS 3.3 System Master Disk), I have found that it helps the readability to number by ten beginning with line 100 and jumping to round numbers for major routines. Renumbering the above program accordingly produces line numbers that go from 100 to 170, then 500 to 570 (see the final listing at the end of this chapter).

It is not necessary for you to use my numbering system, but I would like to stress the importance of having some numbering system that you *consistently* follow whenever possible. Careful attention to small details, like a consistent numbering system and program format, can make your life as a programmer (even as a spare-time programmer) much easier. This attention to detail will also help you develop thinking patterns that aid in creativity and logical solutions to difficult problems.

You now have the knowledge necessary to create a number of large, low-resolution letters. If you change the vertical line of the "I" to the 15th column, you will have a capital "C" (without the rounded edges) (Fig. 6.7). Adding another vertical line in column 25 and extending the two horizontal lines back a couple of columns, you have a capital letter "D" (again in block form) (Fig. 6.8). There are a number of other letters that can be formed by simply moving the horizontal and/or vertical lines. In fact, except for those letters that require a diagonal line, all the other letters can be formed with some combination of HLIN and VLIN commands.

FOR-NEXT Loops

Letters requiring a diagonal are best formed by either plotting every point on the diagonal or devising a routine using available, nongraphic BASIC instructions that tell the computer to repeat certain instructions a specified number of times. These instructions are FOR-NEXT loops. The easiest way to think of FOR-NEXT loops is as one way of telling the

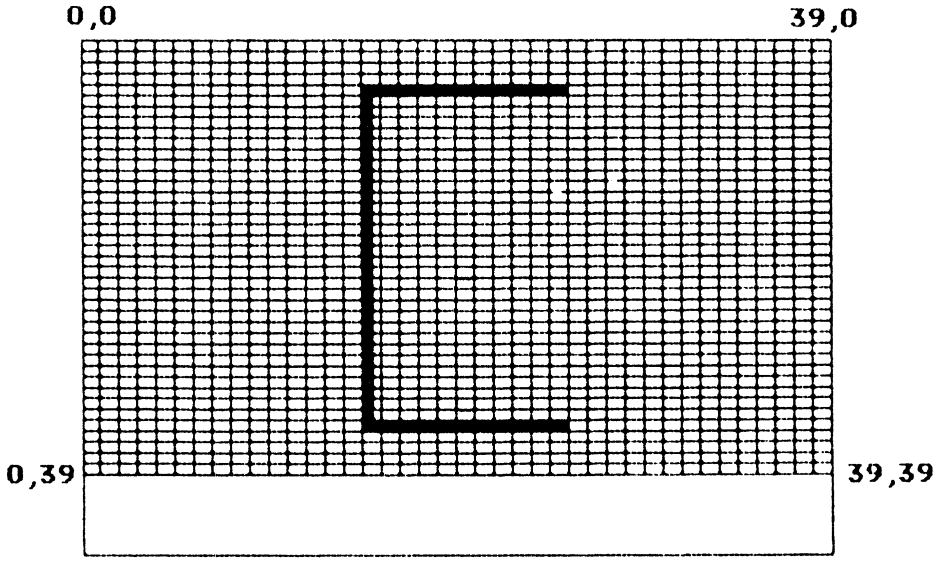


FIGURE 6.7. Low-resolution capital letter C.

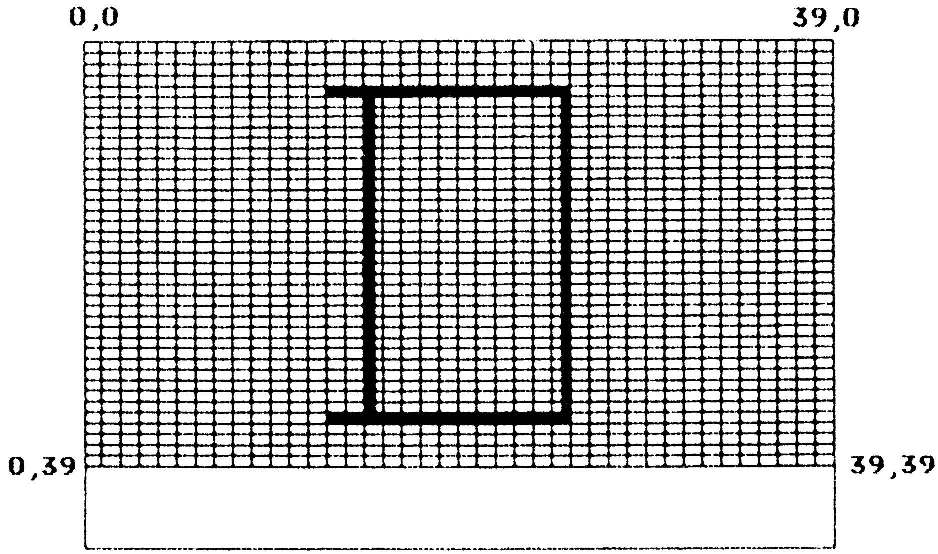


FIGURE 6.8. Low-resolution capital letter D.

computer to count. You instruct the computer to count from some starting number to some ending number, performing a particular instruction at each step. This process usually takes the form:

```
FOR I = 1 TO 10
(an instruction the programmer wants repeated 10 times)
NEXT I
```

For example, if Y = 15 (row 15) and X = 20 (column 20), the instructions

```
FOR I = 1 TO 10
PLOT X + I,Y
NEXT I
```

will draw a horizontal line made up of ten consecutive blocks (from column 20 to column 30) on row 15 (Fig. 6.9).

If the value of Y is also increased each time X is increased, the display should be a diagonal line beginning at the 20th column on the 15th row and ending at the 30th column on the 25th row.

```
X = 20
Y = 15
FOR I = 1 TO 10
PLOT X + I, Y + I
NEXT I
```

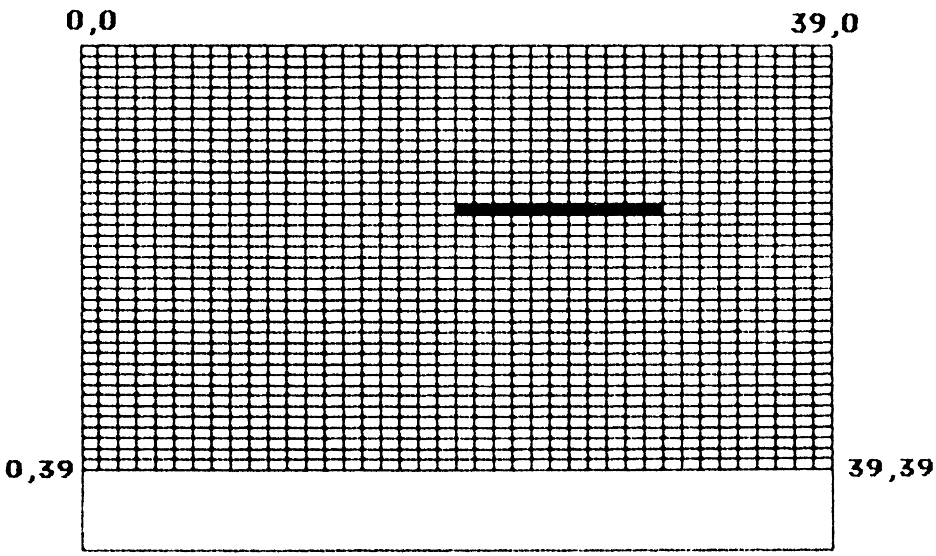


FIGURE 6.9. Low-resolution horizontal line from column 20 to column 30.

Each time the value of the numeric variable I is increased, the computer plots a point one row down and one column over from the last point plotted, producing a diagonal line (Fig. 6.10).

You will need an additional instruction at this point. The STEP command can be added to a FOR-NEXT loop to instruct the computer to count through the loop by the value following the word STEP. In other words, if the instruction is

```
FOR I = 1 TO 10 STEP 2
```

the computer would count from one to ten by twos (i.e., 1, 3, 5, 7, 9). The value after the STEP command can be positive or negative, and it can be a whole number or a decimal value.

With these few commands, the entire alphabet can be formed. In fact, most low-resolution graphics can be produced with the commands covered so far. To demonstrate the capability of these commands, I have included a letter system at the end of this chapter. If you look at the listings for these programs you will see that there are five programs in the system, four of which display letters in some manner and a menu program that allows you to choose between the different programs.

Space does not allow for an explanation of all the code in each of the programs, but I have tried to include enough comments in each program listing so that you can understand what is being done. I urge you to actually read through each of the listings given at the end of this chapter, even if you do not intend to type in the programs. I have found one of the

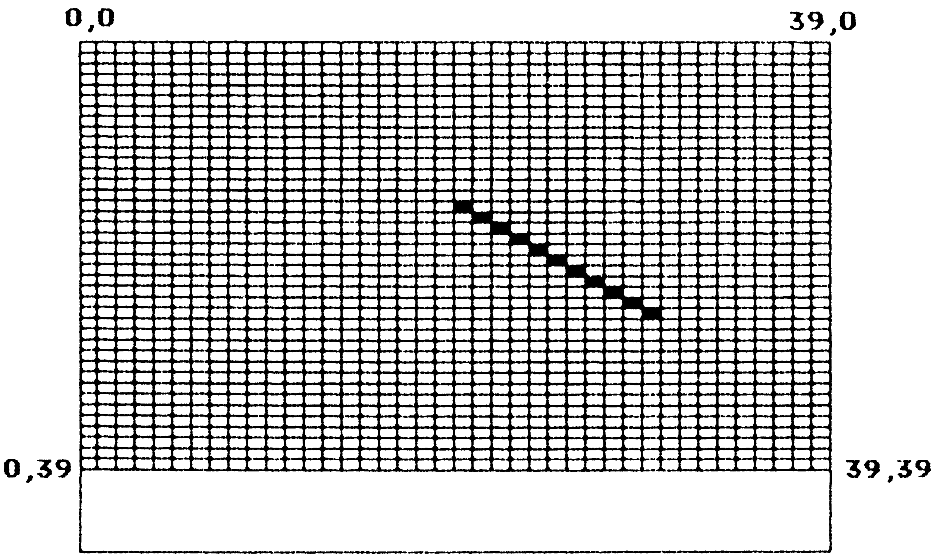


FIGURE 6.10. Low-resolution diagonal line.

best ways to learn programming is by studying well-documented, working programs. When accompanied by some explanation of the overall approach, the documented listings are as instructive as any teacher.

Most of the books on programming include only a portion of the programming necessary to create working systems on the computer. There are some valid reasons for this approach, but I prefer to present a more complete group of listings in the belief that you will benefit from more information in the form of working programs. Ten-to-twenty line programs may be useful for describing specific subjects, but they are not adequate to explain how everything can or should fit together. Snippets of programs also encourage bad programming habits, which eventually cost you more time than is saved by the creation of these quick little programs. Finally, I want to include programs that will be useful to you. Five years after the introduction of my first book, *Apple Files*, people are still using the programs presented in that book. I hope the same will be true for the programs in this book. It may take more time to enter all these programs, but I believe that the time will be worth it and will add considerably to your understanding of the programming process.

EXPLANATION OF LISTINGS

Type Any Letter

The first option in the Letter System Menu is the program TYPE.ANY.LETTER. After providing instructions for the user [please see the program listing at the end of the chapter, instruction lines 170–270 in the TYPE.ANY.LETTER program], the computer waits for a character to be typed on the keyboard [line 410]. When a character is typed, the computer checks to see if the character is a letter using the ASCII code (decimal numbers 65 to 90) [line 440]. (ASCII is the code used in most microcomputers. It sets a decimal and hexadecimal value for each character used.) A check is also made to see if a lowercase letter has been entered. If a lowercase letter has been entered, the computer is instructed to accept that character and convert it to uppercase for display [line 460].

Once the computer has a valid letter, the program switches to the graphic mode and establishes a color selection for the display of the letter [lines 500–550]. A check is made to see which letter was typed [lines 600–626]. After it identifies the letter, the computer is instructed to go to the routine that draws the correct letter [lines 650–901] (Fig. 6.11). After the letter is drawn, the computer returns for another letter [the GOTO 400 instruction included in each letter-drawing routine]. This process continues until the user types the RETURN key. When the RETURN key is pressed, the computer jumps to the routine that ends this program and transfers control to the LETTERS.MENU program [lines 430 and 1000–1020].

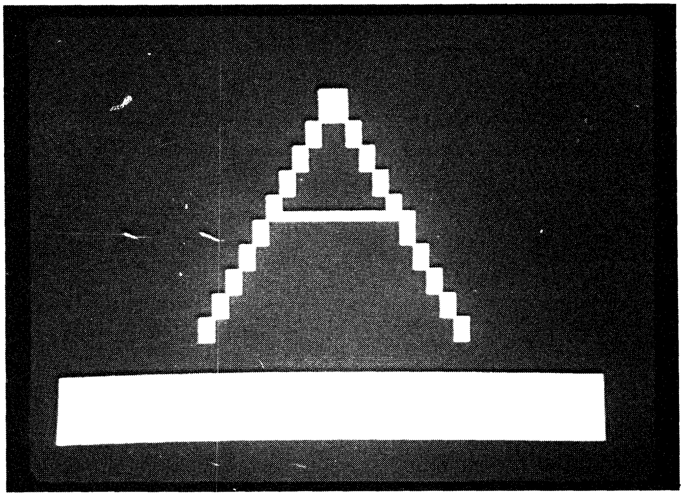


FIGURE 6.11. TYPE.ANY.LETTER program.

Type Displayed Letter

This program is the reverse of the first program. A letter is displayed first [lines 300, 480–901] and the user is to type that letter on the keyboard [line 410]. Once again, when the computer receives a character from the keyboard, that character is checked to see that it is a valid letter (either upper- or lowercase) and then checked to see if the letter typed matches the letter displayed [line 430]. If the two letters do match, the computer goes back and begins the process again [line 480]. If the two letters do not match, the computer returns to wait for the user to type in the correct letter [line 450]. No comment is provided to inform the user that the typed letter is not the displayed letter. The fact that the computer does not display a new letter is the only indication to the user that the incorrect letter was typed. (Some educators may argue that a comment is necessary, but in this circumstance, I feel that none is needed.) Once again, the process continues until the user types the RETURN key [line 420] ending the program. When the RETURN key is pressed, the computer transfers control back to the LETTERS. MENU program [lines 1000–1020].

Display All Letters

This program cycles through the entire alphabet. One after the other, the letter-drawing routines are called according to the letters contained in the data statement [line 2000]. The program can easily be changed to reverse the display of letters by reversing the order of the letters in the data statement. In fact, the letters can be displayed in any order, spelling out words or phrases. The number of letters displayed is not limited either. The display speed can also be easily altered [line 410].

The user cannot end this program with the RETURN key. The entire contents of the data statement (in this case the entire alphabet) must be displayed before control is returned to the LETTERS.MENU program. Programs like this one are used for display purposes or message services.

Type Letter Following Displayed Letter

This program follows the same routine as the TYPE.DISP.LETT program, except that one is added to the ASCII value of the displayed letter [line 410] and the computer checks for the letter that follows the letter displayed on the screen [lines 420, 430, 440 and 460]. If the user types the correct letter (the one that follows the letter displayed alphabetically), the computer goes back and displays another letter [lines 510–901]. Again, if the user does not type the correct letter, the computer simply waits until the correct letter (i.e., the letter that comes after the displayed letter in the alphabet) is typed on the keyboard. As in the first two programs, this program ends when the user types the RETURN key [line 450].

Type Letter Before Displayed Letter

The only difference between this program and the previous one is that one is subtracted from the ASCII value of the letter displayed rather than added to it [line 410]. Subtracting one from the ASCII value of the displayed letter results in the ASCII value for the letter that immediately precedes it in the alphabet. This value is then converted to its equivalent letter [line 430] and the letter is checked to see if it matches the letter entered from the keyboard [line 460]. If the two match, the computer proceeds to display another letter [lines 510–901]. When the user presses the RETURN key, the computer ends the program and returns control to the LETTERS.MENU program.

APPLICATIONS

There are a number of things that can be done to add to this system of programs. Lowercase letters can be added to the display. Letters can be varied in size. More than one letter can be displayed at a time. Letters can be positioned differently and for longer or shorter periods of time. Congratulatory or admonitory messages might be added in response to correct or incorrect keyboard entries. The score of correct responses might be kept and even recorded. Speed of entry can be included as a factor. The MOVING.I program, included at the end of this chapter, suggests a method for varying the size and position of the letters. In that program, the Apple (and on the *II*GS, the option) keys change the size of the "I" while the arrow keys (or Paddles or Joystick) change the location of the "I".

As the programs are written, their use may be adapted to preschoolers learning the alphabet or to adults practicing their typing. With the addition of some of the features mentioned, the LETTERS.SYSTEM program can be specifically designed to suit a variety of applications.

Game Pattern System

The GAME/PATTERN SYSTEM of programs is included to demonstrate a variety of techniques available with low-resolution graphics. The sophistication of the programming varies widely. You are encouraged to carefully study the listings, especially the WALL BLASTER, SWEEPER and MAZE listings.

HOUSE PICTURE

The program was written by a student after one brief lesson using simple PLOT, VLIN and HLIN statements. No sophisticated programming techniques were used. I have included the program to show what can be done with the most elementary programming knowledge. In fact, many things in graphics can be done without the use of more elaborate programming techniques. When the user is finished viewing the house picture (Fig. 6.12), pressing the RETURN key will return control to the GAME.PATT.MENU program.

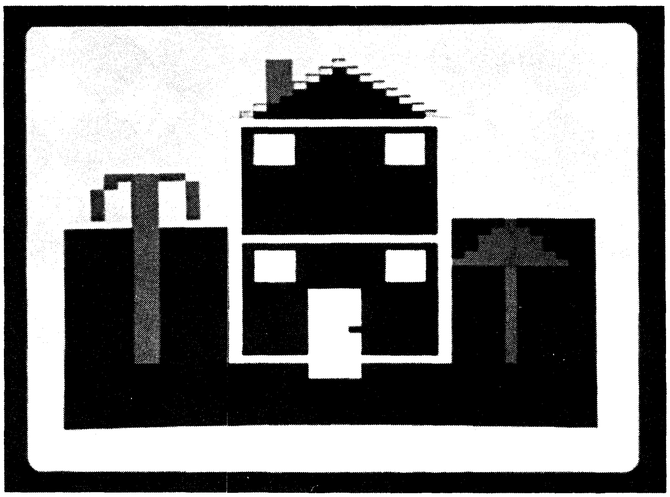


FIGURE 6.12. Low-resolution house picture.

Low-Res Pattern Programs

The pattern programs are considerably more sophisticated than HOUSE.PICTURE, but should be understandable with the help of the many remarks included in the listings. LOWRES.PATTRN.1 divides the screen into five parts and displays patterns in the following order: upper left, lower right, lower left, upper right and finally center pole [lines 1000–5090] (Fig. 6.13). Using FOR-NEXT loops, each section is plotted within its boundaries. The checkerboard effect is achieved with the STEP 2 instruction. The pattern repeats [lines 6000–6030] until the user presses the RETURN key [line 270] and the center pole has finished its pattern. The computer will then return control to the main menu program called GAME.PATT.MENU. The second time the user chooses the LOWRES PATTERN option from the GAME/PATTERN MENU, the computer loads a different program, LOWRES. PATTRN.2. The computer does this by renaming these two programs each time this option is chosen [lines 2703–2800 in the GAME.PATT.MENU program].

LOWRES.PATTRN.2 (Fig. 6.14) does much the same thing as LOWRES.PATTRN.1 except that, after one cycle, the program drops to the BRICK routine [line 6010]. This routine demonstrates the process by which graphic blocks can be moved around the screen. The brick (or ball) actually changes the color of the two sections of the screen involved before beginning to bounce off the walls of the various sections [lines 8000–8420]. Understanding these techniques will allow you to develop even more sophisticated programs that permit the user to control the ball using an input device—the keyboard, a joystick or game paddles.

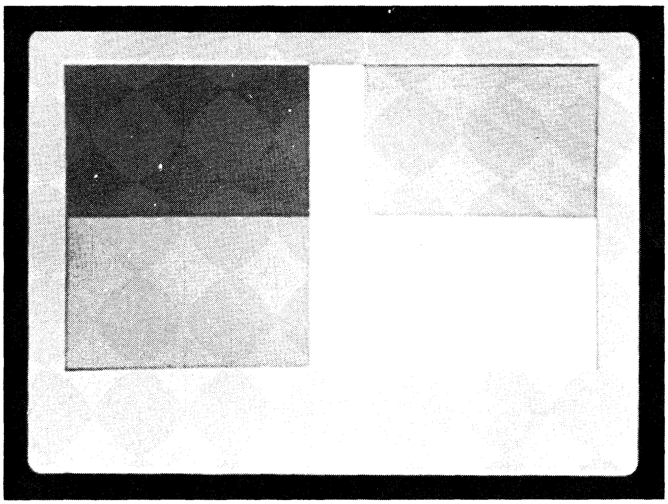


FIGURE 6.13. Low-resolution Pattern 1.

Wall Blaster

The user instructions in this program require considerable programming. As stated in the instructions, this is a game in which the object is "to blast through the walls and explode the square. The last wall canNOT be broken until one of four 'weak' spots is hit. Then any spot in that wall can be broken" and the square can be hit. The user can choose to play the game using either the keyboard, game paddles or a joystick. Instructions for each of these options is included.

The keyboard instructions present a special situation. The Apple *II*e and *II*c keyboards include both Apple keys—the Open Apple and Closed Apple—but the Apple IIGS has only the Open Apple key. The Option key on the IIGS works like the Closed Apple key. Working with a 40-column screen limits the amount of instructions that can or should be included, so no specific instructions are provided for IIGS users.

This program sets up three vertical lines (walls) on the right hand side of the screen [lines 910–960]. Behind these walls, a square is drawn [lines 970–1070—the square is not documented in the program listing]. On the last wall, four weak spots are randomly chosen [lines 1100–1190]. One of these weak spots must be hit before any spot on that last wall can be penetrated (Fig. 6.15). The SCRN command allows you to determine the color of any location on the low-resolution screen. Testing the different wall locations, using the SCRN command, helps determine if one of the walls has been hit.

Once the screen is set up, play begins. The input device (either keyboard, paddles, or joystick) is polled (tested) to detect movement of the Blaster located on the far left side of the screen [lines 2090–2130 for the keyboard and 2160–2260 for either the paddles or joystick]. The computer must also check to see if the Blaster has fired a shot [line 2350]. In addition, the number of shots is stored in the computer's memory [line 2420].

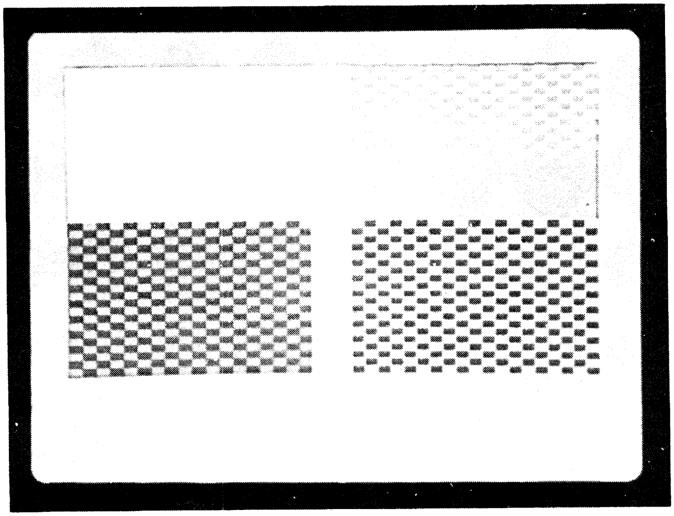


FIGURE 6.14. Low-resolution Pattern 2.

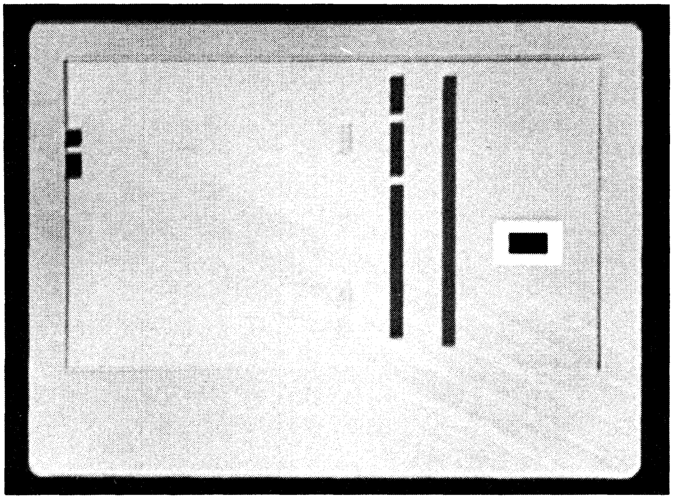


FIGURE 6.15. Wall.Blaster program.

When the user does fire a shot, the shot must cross the screen from the Blaster to a wall. The screen location of the walls must be tested to see if some location on the wall has been hit [lines 2430–2540]. If a location is hit, that location changes color to match the background and therefore becomes a hole in the wall [lines 2480–2530]. When a shot reaches the last wall, the computer checks to see if the shot has struck one of the weak spots [lines 5000–5090]. If it has not hit a weak spot and no weak spot has yet been hit, the computer ignores that shot. If it does hit a weak spot and no weak spot has yet been hit, the computer sets the SECRET$ flag to OFF (indicating that a weak spot has finally been hit) [lines 5010–5050] and then changes the color of that spot to match the background [lines 5120–5130]. Changing the color of that spot indicates that the wall now has a hole in it. From that point on, any hits on the wall will produce a hole in the wall. Subsequent shots through the holes can attempt to hit the square.

After every shot, the square moves by being erased from its current position and replotted at its new, randomly chosen, position [lines 7000–7220]. When one of the shots finally hits the square, the square must explode and the computer's bell sounds [lines 8000–8640]. The user is then shown the number of shots it took to explode the square and asked if the user wishes to play again [lines 9000–9100]. If the user indicates that he or she wants to play again, the program begins again [line 9120]. If the indication is that the user does not want to play again, the computer returns control to the GAME.PATT.MENU program [lines 9110 and 10000–10060].

The WALL.BLASTER program is a fairly complicated program with a number of things going on at the same time. The listing is quite heavily documented with remarks that help clarify some of the more obscure instructions. A careful study of this program should lead to a thorough understanding of low-resolution graphics and the various programming techniques possible with Applesoft BASIC.

Sweeper and Maze

These programs use many of the same techniques demonstrated in the WALL.BLASTER game. The SWEEPER game uses a vacuum to sweep up all the spots that have been created by the program [lines 1500–1600] (Fig. 6.16). This game does not check to see if all the spots have been cleared but allows the user to press the ESC key at any time and end the game [lines 2150, 2200 and 2420]. Obviously, additions to this game would be to have the computer check to see if all the spots have been vacuumed and to have the computer automatically end when all the spots are cleared from the screen.

The sweeper program has one other feature not demonstrated in any of the other programs in this chapter. SWEEPER allows the user to turn the sound on and off [lines 2490, 2510, 3010 and 3020]. One method of producing sound is demonstrated in the WALL.BLASTER program and the MAZE game, but only on the occasion of a specific event (i.e., the square exploding). In SWEEPER, the sound (supposedly the sound of the vacuum) is on all the time [lines 6000–6030]. However, there are many situations when sound may not be appropriate (i.e., in a classroom, late at night). Therefore, I included a method that allows the user to choose to eliminate the sound. The technique presented in this program can be used in many other situations to allow the user more control over the various elements involved in application programs.

The MAZE game is included to demonstrate that a low-resolution game can be very challenging for a wide variety of users with varying levels of dexterity (Fig. 6.17). The game was created in such a way that, in its most difficult mode (the joystick mode) the user can still win, and in its easiest mode (the keyboard mode with the Open Apple key held down), the user is still challenged.

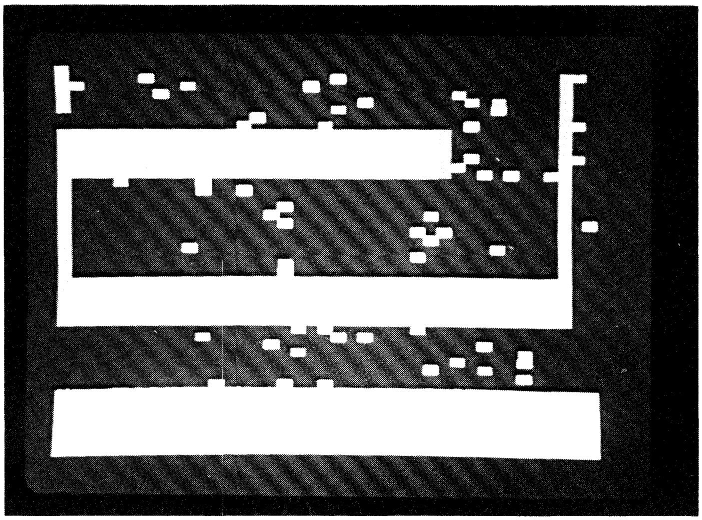


FIGURE 6.16. Sweeper program.

Free Drawing and View Drawing

Both of these programs were intentionally left in a primitive state. They are incomplete programs, and you are encouraged to improve on and finish them. Some suggestions for additions to these programs include:

- User instructions for keyboard, joystick and/or paddle input as well as routines that allow for all these forms of input
- Routines that allow the user to setup or recall a drawing before adding to the drawing
- Routines that allow editing of current or past drawings
- Improved error-checking routines

There are many other features that can be added to these drawing programs, and a number of the techniques presented in this chapter should provide you with the ability to alter and improve these incomplete programs in just about any way you want.

There is one problem in the programs that currently seems insurmountable. The VIEW.DRAWING program works fine under DOS 3.3 but does not work under any version of ProDOS. The FREE.DRAWING program allows the user to save a copy of the drawing on the screen by dumping the contents of Text/Low-Resolution Page 1 to a binary file. The VIEW.DRAWING program then loads that binary file from diskette and, under DOS 3.3, the drawing is displayed. But ProDOS apparently uses some of the memory locations in Text/Low-Resolution Page 1 (memory pages four through seven) and does not allow the entire contents of those memory locations to be replaced. Whenever you attempt to load the binary file under ProDOS, an error message is displayed indicating that there are "NO BUFFERS AVAILABLE." After extensive effort and research, I have concluded that

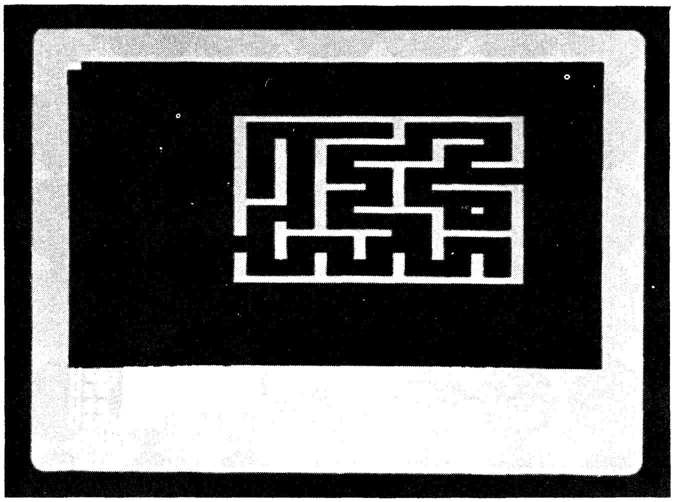


FIGURE 6.17. Maze program.

the time it would take to *possibly* make this program work under ProDOS is simply not worth it. If you would like to save and reload low-resolution information, do so under DOS 3.3 rather than ProDOS.

To conclude this chapter, I would like to point out that most of the techniques used to create low-resolution programs can be used to create similar high-resolution programs. This is one of the reasons I have taken so much space to deal with low-resolution graphics. Because so much procedural information is involved in high-resolution graphics, programming techniques and routines are left out or covered briefly in many books. Having presented the techniques and routines in the low-resolution graphic section, I will be free to cover the procedural information in the high-resolution chapters without overwhelming you and, at the same time, I hope I have demonstrated that low-resolution is a viable option for certain situations.

But you can only apply these techniques and routines to high-resolution graphics if you have fully read and understood them. Once again, I urge you to read and study the program listings at the end of the chapter. The listings are intended to be a part of the text and are documented by many, many remarks that explain what certain sections accomplish or what individual commands do. Not every line in every program is documented, but every obscure or unclear instruction is documented somewhere in these program listings. Explaining every instruction in the body of this chapter would have been redundant and would have vastly increased the size of an already long chapter.

REVIEW QUESTIONS

1. What does the HLIN command do?
2. What does the VLIN command do?
3. What does the PLOT command do?
4. What does the SCRN command do?
5. True or False? In order to create low-resolution graphics, the GR (or equivalent) command must be issued and a color established before any information can be plotted.
6. What is the range of ASCII values for capital letters, A–Z?
7. What command is used to tell the computer to wait for a key to be entered from the keyboard?
8. What command is used to clear the keyboard buffer?

PRODOS NOTE

For those using the ProDOS operating system, I include below the instructions necessary to place the different program systems into subdirectories. These instructions are listed separately from the regular program listings to avoid confusing those not using ProDOS or not using subdirectories.

In each of the menu programs, include the following code:

```
142 YES = PEEK(800)
143 IF YES = 244 THEN 170
145 PRINT D$; "PREFIX LETTERS.FOLD"
147 POKE 800,244: REM SET FLAG
```

For the GAME/PATTERN SYSTEM programs, line 145 should read:

```
145 PRINT D$; "PREFIX GAME.PATT.FOLD"
```

If you want to include the LOW.HIGH.COMPARISON SYSTEM programs from Chapter 5 in a subdirectory, line 145 should read:

```
145 PRINT D$; "PREFIX LOW.HIGH.FOLD"
```

Two other lines should be included in each menu program. These instructions are the same in each menu program but require different line numbers.

For the LETTERS.MENU program:

```
7045 POKE 800,0:REM RESET FLAG
7055 PRINT D$; "PREFIX ,S5,D1"
```

For the GAME.PATT.MENU program the line numbers are 9045 and 9055. For the LOW.HIGH.MENU program the line numbers are 8045 and 8055.

Low-Res Capital "I"

```
100  REM  ***--LOW.RES.I--***
110 :
120 :
130  REM  **--INITIALIZE LOW-RES GRAPHICS--**
140  GR
150  COLOR= 6: REM  SET COLOR TO BLUE
160 :
170 :
500  REM  **--DRAW CAPITAL LETTER I--**
510  VLIN 5,35, AT 20
520  HLIN 15,25 AT 5
530  HLIN 15,25 AT 35
540 :
550 :
560  REM  **--END PROGRAM--**
570  END
```

Letters Menu

```
100  REM ***--LETTERS.MENU--***
110 :
120 :
130 TB = 8
140 D$ =  CHR$ (4): REM CONTROL D
150 :
160 :
170  REM **--DISPLAY MENU--**
180  TEXT : HOME
190  VTAB 3
200  HTAB TB + 4
210  INVERSE
220  PRINT "LETTERS MENU"
230  NORMAL
240  PRINT : PRINT
250  HTAB TB
260  PRINT "1.  TYPE ANY LETTER"
270  PRINT : HTAB TB
280  PRINT "2.  TYPE DISPLAYED LETTER"
290  PRINT : HTAB TB
```

```
300  PRINT "3.  DISPLAY OF ALL LETTERS"
310  PRINT : HTAB TB
320  PRINT "4.  TYPE LETTER THAT FOLLOWS"
330  PRINT : HTAB TB
340  PRINT "5.  TYPE LETT.THAT COMES BEFORE"
350  PRINT : HTAB TB
360  PRINT "6.  CATALOG"
370  PRINT : HTAB TB
380  PRINT "7.  END"
390  PRINT : HTAB TB
400  PRINT "WHICH NUMBER PLEASE? (";: INVERSE : PRINT "1-7";: NORMAL : PRINT
    "):";
410  GET NB$: PRINT
420 NB =  VAL (NB$)
430  IF NB < 1 OR NB > 7 THEN  PRINT : HTAB TB: INVERSE : PRINT "INCORRE
    CT CHOICE!": NORMAL : GOTO 190
440  IF NB = 1 THEN 1000
450  IF NB = 2 THEN 2000
460  IF NB = 3 THEN 3000
470  IF NB = 4 THEN 4000
480  IF NB = 5 THEN 5000
490  IF NB = 6 THEN 6000
500  IF NB = 7 THEN 7000
510 :
520 :
1000  REM **--TYPE ANY LETTER--**
1010  PRINT D$;"RUN TYPE.ANY.LETTER"
1980 :
1990 :
2000  REM **--TYPE DISPLAYED LETTER--**
2010  PRINT D$;"RUN TYPE.DISP.LETT"
2980 :
2999 :
3000  REM **--DISPLAY ALL LETTERS--**
3010  PRINT D$;"RUN DISP.ALL.LETT"
3980 :
3990 :
4000  REM **--TYPE LETTER FOLLOWING DISPLAYED LETTER--**
4010  PRINT D$;"RUN TYPE.LETT.FOLL"
4980 :
4990 :
5000  REM **--TYPE LETTER THAT COMES BEFORE DISPLAYED LETTER--**
5010  PRINT D$;"RUN TYPE.LET.BEFORE"
5980 :
5990 :
6000  REM **--DISPLAY CATALOG--**
6010  HOME
```

```
6020  PRINT D$;"CATALOG"
6030  PRINT
6040  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
6050  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT " K
    EY TO CONTINUE:";L$
6060  GOTO 170
6980 :
6990 :
7000  REM   **--END--**
7010  HOME
7020  VTAB 10
7030  INVERSE
7040  PRINT "SEE YOU NEXT TIME."
7050  NORMAL
7060  END
9998 :
9999 :
10000  REM  LETTERS SYSTEM
11000  REM  COPYRIGHT OCTOBER 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS RD
14000  REM  CORVALLIS OR. 97330-9340
```

Type Any Letter

```
100  REM ***--TYPE.ANY.LETTER--***
110 :
120 :
130 D$ =  CHR$ (4): REM CONTROL D
140  POKE  - 16368,0: REM CLEAR KBRD BUFFER
150 :
160 :
170  REM **--USER INSTRUCTIONS--**
180  HOME
190  INVERSE : HTAB 13: PRINT "INSTRUCTIONS": NORMAL
200  VTAB 8
210  PRINT "PRESS ANY LETTER KEY TO SEE"
220  PRINT
230  PRINT "A LARGE DISPLAY OF THAT LETTER."
240  VTAB 15
250  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT " KE
    Y TO END THE PROGRAM."
260  PRINT
```

```
270  PRINT : PRINT : PRINT "PRESS ANY LETTER KEY (";: INVERSE : PRINT "A
    -Z";: NORMAL : PRINT ") TO BEGIN. ";
280 :
290 :
400  REM **--GET LETTER--**
410  GET LTR$
420  PRINT
430  IF  ASC (LTR$) = 13 THEN  TEXT : HOME : GOTO 1000
440  IF  ASC (LTR$) < 65 OR  ASC (LTR$) > 90 THEN 460
450  GOTO 500: REM SKIP LOWER CASE TEST
460  IF  ASC (LTR$) > 96 AND  ASC (LTR$) < 123 THEN LTR$ = CHR$ ( ASC (
    LTR$) - 32): GOTO 500
470  GOTO 410: REM INVALID ENTRY TRY AGAIN
480 :
490 :
500  REM **--DISPLAY LETTER--**
510  GR
520 CLR =  INT ( RND (1) * 100)
530  IF CLR = 0 THEN 520
540  IF CLR > 15 THEN 520
550  COLOR= CLR
560 :
570 :
600  REM **--GOTO DRAWING ROUTINE--**
601  IF LTR$ = "A" THEN 650
602  IF LTR$ = "B" THEN 660
603  IF LTR$ = "C" THEN 670
604  IF LTR$ = "D" THEN 680
605  IF LTR$ = "E" THEN 690
606  IF LTR$ = "F" THEN 700
607  IF LTR$ = "G" THEN 710
608  IF LTR$ = "H" THEN 720
609  IF LTR$ = "I" THEN 730
610  IF LTR$ = "J" THEN 740
611  IF LTR$ = "K" THEN 750
612  IF LTR$ = "L" THEN 760
613  IF LTR$ = "M" THEN 770
614  IF LTR$ = "N" THEN 780
615  IF LTR$ = "O" THEN 790
616  IF LTR$ = "P" THEN 800
617  IF LTR$ = "Q" THEN 810
618  IF LTR$ = "R" THEN 820
619  IF LTR$ = "S" THEN 830
620  IF LTR$ = "T" THEN 840
621  IF LTR$ = "U" THEN 850
622  IF LTR$ = "V" THEN 860
```

```
623  IF LTR$ = "W" THEN 870
624  IF LTR$ = "X" THEN 880
625  IF LTR$ = "Y" THEN 890
626  IF LTR$ = "Z" THEN 900
627 :
628 :
650  REM **--A--**
651 X = 20: FOR Y = 5 TO 35: PLOT X,Y:X = X - .33: NEXT Y:X = 20: FOR Y =
    5 TO 35: PLOT X,Y:X = X + .33: NEXT Y: HLIN 15,25 AT 20: PLOT 19,5: GOTO
    400
658 :
659 :
660  REM **--B--**
661  VLIN 5,35 AT 15: HLIN 13,25 AT 5: VLIN 5,20 AT 25: HLIN 17,25 AT 20
    : HLIN 17, 25 AT 20: VLIN 21,35 AT 25: HLIN 13,25 AT 35: GOTO 400
668 :
669 :
670  REM **--C--**
671  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,25 AT 35: GOTO 400
678 :
679 :
680  REM **--D--**
681  VLIN 5,35 AT 15: HLIN 13,25 AT 5: VLIN 5,35 AT 25: HLIN 13,25 AT 35
    : GOTO 400
688 :
689 :
690  REM **--E--**
691  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,20 AT 20: HLIN 15,25 AT 3
    5: GOTO 400
698 :
699 :
700  REM **--F--**
701  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,20 AT 20: GOTO 400
708 :
709 :
710  REM **--G--**
711  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,25 AT 35: VLIN 25,35 AT 2
    5: HLIN 20,25 AT 25: GOTO 400
718 :
719 :
720  REM **--H--**
721  VLIN 5,35 AT 15: VLIN 5,35 AT 25: HLIN 15,25 AT 20: GOTO 400
728 :
729 :
730  REM **--I--**
731  VLIN 5,35 AT 20: HLIN 15,25 AT 5: HLIN 15,25 AT 35: GOTO 400
```

```
738 :
739 :
740  REM **--J--**
741  VLIN 5,35 AT 25: HLIN 15,25 AT 35: VLIN 30,35 AT 15: HLIN 20,30 AT
    5: GOTO 400
748 :
749 :
750  REM **--K--**
751  VLIN 5,35 AT 15:X = 20: FOR Y = 15 TO 35: PLOT X,Y:X = X + .5: NEXT
    Y:Y = 20: FOR X = 15 TO 28: PLOT X,Y:Y = Y - 1: NEXT X: GOTO 400
758 :
759 :
760  REM **--L--**
761  VLIN 5,35 AT 15: HLIN 15,25 AT 35: GOTO 400
768 :
769 :
770  REM **--M--**
771  VLIN 6,35 AT 11: VLIN 6,35 AT 30
772 X = 11:Y = 6
773  PLOT X,Y
774 X = X + 1:Y = Y + 1
775  IF X = 21 AND Y = 16 THEN 777
776  GOTO 773
777 X = 21: FOR Y = 15 TO 6 STEP  - 1: PLOT X,Y:X = X + 1: NEXT Y: GOTO
    400
778 :
779 :
780  REM **--N--**
781  VLIN 5,34 AT 14:X = 15: FOR Y = 5 TO 34: PLOT X,Y:X = X + .5: NEXT
    Y: VLIN 5,34 AT 30: GOTO 400
788 :
789 :
790  REM **--O--**
791  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,35 AT 25: HLIN 15,25 AT 35
    : GOTO 400
798 :
799 :
800  REM **--P--**
801  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,20 AT 25: HLIN 15,25 AT 20
    : GOTO 400
808 :
809 :
810  REM **--Q--**
811  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,35 AT 25: HLIN 15,25 AT 35
    : PLOT 26,36: PLOT 27,37: PLOT 28,38: PLOT 29,39: GOTO 400
818 :
```

```
819 :
820  REM **--R--**
821  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,15 AT 25: HLIN 15,25 AT 15
    :X = 18: FOR Y = 16 TO 35: PLOT X,Y:X = X + .5: NEXT Y: GOTO 400
828 :
829 :
830  REM **--S--**
831  HLIN 15,25 AT 5: VLIN 5,20 AT 15: HLIN 15,25 AT 20: VLIN 21,35 AT 2
    5: HLIN 15,25 AT 35: GOTO 400
838 :
839 :
840  REM **--T--**
841  HLIN 13,28 AT 5: VLIN 5,35 AT 20: GOTO 400
848 :
849 :
850  REM **--U--**
851  VLIN 5,35 AT 15: HLIN 15,25 AT 35: VLIN 5,35 AT 25: GOTO 400
858 :
859 :
860  REM **--V--**
861 X = 10: FOR Y = 5 TO 35: PLOT X,Y:X = X + .33: NEXT Y:X = 30: FOR Y =
    5 TO 35: PLOT X,Y:X = X - .33: NEXT Y: GOTO 400
868 :
869 :
870  REM **--W--**
871  VLIN 6,35 AT 10: VLIN 6,35 AT 30:X = 10: FOR Y = 35 TO 16 STEP  - 1
    : PLOT X,Y:X = X + .5: NEXT Y:X = 21: FOR Y = 16 TO 35: PLOT X,Y:X =
    X + .5: NEXT Y: PLOT 20,15: GOTO 400
878 :
879 :
880  REM **--X--**
881 X = 13: FOR Y = 5 TO 35: PLOT X,Y:X = X + .5: NEXT Y:X = 28: FOR Y =
    5 TO 35: PLOT X,Y:X = X - .5: NEXT Y: GOTO 400
888 :
889 :
890  REM **--Y--**
891  VLIN 19,39 AT 20:X = 8: FOR Y = 6 TO 18: PLOT X,Y:X = X + 1: NEXT Y
    :X = 32: FOR Y = 6 TO 18: PLOT X,Y:X = X - 1: NEXT Y: GOTO 400
898 :
899 :
900  REM **--Z--**
901  HLIN 15,30 AT 5: HLIN 15,30 AT 36:X = 30: FOR Y = 6 TO 35: PLOT X,Y
    :X = X - .5: NEXT Y: GOTO 400
998 :
999 :
1000  REM **--RETURN TO LETTERS MENU--**
```

```
1010  PRINT
1020  PRINT D$;"RUN LETTERS.MENU"
```

Type Displayed Letter

```
100  REM ***--TYPE.DISP.LETT--***
110 :
120 :
130 D$ =  CHR$ (4): REM CONTROL D
140  POKE  - 16368,0: REM CLEAR KBRD BUFFER
150 :
160 :
170  REM **--USER INSTRUCTIONS--**
180  HOME
190  INVERSE : HTAB 13: PRINT "INSTRUCTIONS": NORMAL
200  VTAB 8
210  PRINT "PRESS THE CORRECT KEY FOR THE"
220  PRINT
230  PRINT "LETTER DISPLAYED ON THE SCREEN."
240  VTAB 15
250  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT " KE
    Y TO END THE PROGRAM."
260  PRINT
265  PRINT : PRINT
270  PRINT "PRESS THE ";: INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT "
     TO BEGIN.";
280  GET L$: PRINT
290  IF  ASC (L$) <  > 32 THEN 240
300  GOTO 480: REM  DISPLAY LETTER ROUTINE
310 :
320 :
400  REM **--GET LETTER--**
410  GET LR$
420  IF  ASC (LR$) = 13 THEN  TEXT : HOME : GOTO 1000
430  IF LR$ = LTR$ THEN 480
440  IF  ASC (LR$) > 96 AND  ASC (LR$) < 123 THEN LR$ = CHR$ ( ASC (LR$
    ) - 32): GOTO 430: REM TEST AGAIN
450  GOTO 410: REM INVALID ENTRY TRY AGAIN
460 :
470 :
480  REM **--DISPLAY LETTER--**
490  GR
500 CLR =  INT ( RND (1) * 100)
510  IF CLR = 0 THEN 500
```

```
520  IF CLR > 15 THEN 500
530  COLOR= CLR
540 LT =  INT ( RND (1) * 100)
550  IF LT < 65 OR LT > 90 THEN 540
560 LTR$ = CHR$ (LT)
570 :
580 :
600  REM **--GOTO DRAWING ROUTINE--**
601  IF LTR$ = "A" THEN 650
602  IF LTR$ = "B" THEN 660
603  IF LTR$ = "C" THEN 670
604  IF LTR$ = "D" THEN 680
605  IF LTR$ = "E" THEN 690
606  IF LTR$ = "F" THEN 700
607  IF LTR$ = "G" THEN 710
608  IF LTR$ = "H" THEN 720
609  IF LTR$ = "I" THEN 730
610  IF LTR$ = "J" THEN 740
611  IF LTR$ = "K" THEN 750
612  IF LTR$ = "L" THEN 760
613  IF LTR$ = "M" THEN 770
614  IF LTR$ = "N" THEN 780
615  IF LTR$ = "O" THEN 790
616  IF LTR$ = "P" THEN 800
617  IF LTR$ = "Q" THEN 810
618  IF LTR$ = "R" THEN 820
619  IF LTR$ = "S" THEN 830
620  IF LTR$ = "T" THEN 840
621  IF LTR$ = "U" THEN 850
622  IF LTR$ = "V" THEN 860
623  IF LTR$ = "W" THEN 870
624  IF LTR$ = "X" THEN 880
625  IF LTR$ = "Y" THEN 890
626  IF LTR$ = "Z" THEN 900
627 :
628 :
650  REM **--A--**
651 X = 20: FOR Y = 5 TO 35: PLOT X,Y:X = X - .33: NEXT Y:X = 20: FOR Y =
    5 TO 35: PLOT X,Y:X = X + .33: NEXT Y: HLIN 15,25 AT 20: PLOT 19,5: GOTO
    400
658 :
659 :
660  REM **--B--**
661  VLIN 5,35 AT 15: HLIN 13,25 AT 5: VLIN 5,20 AT 25: HLIN 17,25 AT 20
    : HLIN 17,25 AT 20: VLIN 21,35 AT 25: HLIN 13,25 AT 35: GOTO 400
668 :
```

```
669 :
670  REM **--C--**
671  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,25 AT 35: GOTO 400
678 :
679 :
680  REM **--D--**
681  VLIN 5,35 AT 15: HLIN 13,25 AT 5: VLIN 5,35 AT 25: HLIN 13,25 AT 35
    : GOTO 400
688 :
689 :
690  REM **--E--**
691  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,20 AT 20: HLIN 15,25 AT 35:
    GOTO 400
698 :
699 :
700  REM **--F--**
701  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,20 AT 20: GOTO 400
708 :
709 :
710  REM **--G--**
711  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,25 AT 35: VLIN 25,35 AT 25:
    HLIN 20,25 AT 25: GOTO 400
718 :
719 :
720  REM **--H--**
721  VLIN 5,35 AT 15: VLIN 5,35 AT 25: HLIN 15,25 AT 20: GOTO 400
728 :
729 :
730  REM **--I--**
731  VLIN 5,35 AT 20: HLIN 15,25 AT 5: HLIN 15,25 AT 35: GOTO 400
738 :
739 :
740  REM **--J--**
741  VLIN 5,35 AT 25: HLIN 15,25 AT 35: VLIN 30,35 AT 15: HLIN 20,30 AT
    5: GOTO 400
748 :
749 :
750  REM **--K--**
751  VLIN 5,35 AT 15:X = 20: FOR Y = 15 TO 35: PLOT X,Y:X = X + .5: NEXT
    Y:Y = 20: FOR X = 15 TO 28: PLOT X,Y:Y = Y - 1: NEXT X: GOTO 400
758 :
759 :
760  REM **--L--**
761  VLIN 5,35 AT 15: HLIN 15,25 AT 35: GOTO 400
768 :
769 :
```

```
770  REM **--M--**
771  VLIN 6,35 AT 11: VLIN 6,35 AT 30
772 X = 11:Y = 6
773  PLOT X,Y
774 X = X + 1:Y = Y + 1
775  IF X = 21 AND Y - 16 THEN 777
776  GOTO 773
777 X = 21: FOR Y = 15 TO 6 STEP  - 1: PLOT X,Y:X = X + 1: NEXT Y: GOTO
    400
778 :
779 :
780  REM **--N--**
781  VLIN 5,34 AT 14:X = 15: FOR Y = 5 TO 34: PLOT X,Y:X = X + .5: NEXT
    Y: VLIN 5,34 AT 30: GOTO 400
788 :
789 :
790  REM **--O--**
791  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,35 AT 25: HLIN 15,25 AT 35
    : GOTO 400
798 :
799 :
800  REM **--P--**
801  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,20 AT 25: HLIN 15,25 AT 20
    : GOTO 400
808 :
809 :
810  REM **--Q--**
811  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,35 AT 25: HLIN 15,25 AT 35
    : PLOT 26,36: PLOT 27,37: PLOT 28,38: PLOT 29,39: GOTO 400
818 :
819 :
820  REM **--R--**
821  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,15 AT 25: HLIN 15,25 AT 15
    :X = 18: FOR Y = 16 TO 35: PLOT X,Y:X = X + .5: NEXT Y: GOTO 400
828 :
829 :
830  REM **--S--**
831  HLIN 15,25 AT 5: VLIN 5,20 AT 15: HLIN 15,25 AT 20: VLIN 21,35 AT 2
    5: HLIN 15,25 AT 35: GOTO 400
838 :
839 :
840  REM **--T--**
841  HLIN 13,28 AT 5: VLIN 5,35 AT 20: GOTO 400
848 :
849 :
850  REM **--U--**
```

```
851  VLIN 5,35 AT 15: HLIN 15,25 AT 35: VLIN 5,35 AT 25: GOTO 400
858 :
859 :
860  REM **--V--**
861 X = 10: FOR Y = 5 TO 35: PLOT X,Y:X = X + .33: NEXT Y:X = 30: FOR Y =
    5 TO 35: PLOT X,Y:X = X - .33: NEXT Y: GOTO 400
868 :
869 :
870  REM **--W--**
871  VLIN 6,35 AT 10: VLIN 6,35 AT 30:X = 10: FOR Y = 35 TO 16 STEP  - 1
    : PLOT X,Y:X = X + .5: NEXT Y:X = 21: FOR Y = 16 TO 35: PLOT X,Y:X =
    X + .5: NEXT Y: PLOT 20,15: GOTO 400
878 :
879 :
880  REM **--X--**
881 X = 13: FOR Y = 5 TO 35: PLOT X,Y:X = X + .5: NEXT Y:X = 28: FOR Y =
    5 TO 35: PLOT X,Y:X = X - .5: NEXT Y: GOTO 400
888 :
889 :
890  REM **--Y--**
891  VLIN 19,39 AT 20:X = 8: FOR Y = 6 TO 18: PLOT X,Y:X = X + 1: NEXT Y
    :X = 32: FOR Y = 6 TO 18: PLOT X,Y:X = X - 1: NEXT Y: GOTO 400
898 :
899 :
900  REM **--Z--**
901  HLIN 15,30 AT 5: HLIN 15,30 AT 36:X = 30: FOR Y = 6 TO 35: PLOT X,Y
    :X = X - .5: NEXT Y: GOTO 400
998 :
999 :
1000  REM **--RETURN TO LETTERS MENU--**
1010  PRINT
1020  PRINT D$;"RUN LETTERS.MENU"
```

Display All Letters

```
100  REM ***--DISP.ALL.LETT--***
110 :
120 :
130 D$ =  CHR$ (4): REM CONTROL D
140  POKE  - 16368,0: REM CLEAR KBRD BUFFER
150 :
160 :
170  REM **--USER INSTRUCTIONS--**
180  HOME
190  INVERSE : HTAB 13: PRINT "INSTRUCTIONS": NORMAL
```

```
200  VTAB 8
210  PRINT "WATCH THE ALPHABET GO BY!"
220  PRINT : PRINT
230  PRINT "YOU DO NOT HAVE TO PRESS ANY KEY"
240  PRINT : PRINT
250  PRINT "EXCEPT TO START!"
260  VTAB 18
265  PRINT : PRINT
270  PRINT "PRESS THE ";: INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT "
     TO BEGIN.";
280  GET L$: PRINT
290  IF  ASC (L$) <  > 32 THEN 260
300 :
310 :
400  REM **--GET LETTER--**
410  FOR I = 1 TO 1000: NEXT I
420  READ LTR$
430  IF LTR$ = "END" THEN  TEXT : HOME : GOTO 1000
440 :
450 :
460  REM **--DISPLAY LETTER--**
470  GR
480 CLR =  INT ( RND (1) * 100)
490  IF CLR = 0 THEN 480
500  IF CLR > 15 THEN 480
510  COLOR= CLR
520 :
530 :
600  REM **--GOTO DRAWING ROUTINE--**
601  IF LTR$ = "A" THEN 650
602  IF LTR$ = "B" THEN 660
603  IF LTR$ = "C" THEN 670
604  IF LTR$ = "D" THEN 680
605  IF LTR$ = "E" THEN 690
606  IF LTR$ = "F" THEN 700
607  IF LTR$ = "G" THEN 710
608  IF LTR$ = "H" THEN 720
609  IF LTR$ = "I" THEN 730
610  IF LTR$ = "J" THEN 740
611  IF LTR$ = "K" THEN 750
612  IF LTR$ = "L" THEN 760
613  IF LTR$ = "M" THEN 770
614  IF LTR$ = "N" THEN 780
615  IF LTR$ = "O" THEN 790
616  IF LTR$ = "P" THEN 800
617  IF LTR$ = "Q" THEN 810
```

```
618  IF LTR$ = "R" THEN 820
619  IF LTR$ = "S" THEN 830
620  IF LTR$ = "T" THEN 840
621  IF LTR$ = "U" THEN 850
622  IF LTR$ = "V" THEN 860
623  IF LTR$ = "W" THEN 870
624  IF LTR$ = "X" THEN 880
625  IF LTR$ = "Y" THEN 890
626  IF LTR$ = "Z" THEN 900
627 :
628 :
650  REM **--A--**
651 X = 20: FOR Y = 5 TO 35: PLOT X,Y:X = X - .33: NEXT Y:X = 20: FOR Y =
    5 TO 35: PLOT X,Y:X = X + .33: NEXT Y: HLIN 15,25 AT 20: PLOT 19,5: GOTO
    400
658 :
659 :
660  REM **--B--**
661  VLIN 5,35 AT 15: HLIN 13,25 AT 5: VLIN 5,20 AT 25: HLIN 17,25 AT 20
    : HLIN 17,25 AT 20: VLIN 21,35 AT 25: HLIN 13,25 AT 35: GOTO 400
668 :
669 :
670  REM **--C--**
671  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,25 AT 35: GOTO 400
678 :
679 :
680  REM **--D--**
681  VLIN 5,35 AT 15: HLIN 13,25 AT 5: VLIN 5,35 AT 25: HLIN 13,25 AT 35:
    GOTO 400
688 :
689 :
690  REM **--E--**
691  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,20 AT 20: HLIN 15,25 AT 3
    5: GOTO 400
698 :
699 :
700  REM **--F--**
701  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,20 AT 20: GOTO 400
708 :
709 :
710  REM **--G--**
711  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,25 AT 35: VLIN 25,35 AT 2
    5: HLIN 20,25 AT 25: GOTO 400
718 :
719 :
720  REM **--H--**
```

```
721  VLIN 5,35 AT 15: VLIN 5,35 AT 25: HLIN 15,25 AT 20: GOTO 400
728 :
729 :
730  REM **--I--**
731  VLIN 5,35 AT 20: HLIN 15,25 AT 5: HLIN 15,25 AT 35: GOTO 400
738 :
739 :
740  REM **--J--**
741  VLIN 5,35 AT 25: HLIN 15,25 AT 35: VLIN 30,35 AT 15: HLIN 20,30 AT
    5: GOTO 400
748 :
749 :
750  REM **--K--**
751  VLIN 5,35 AT 15:X = 20: FOR Y = 15 TO 35: PLOT X,Y:X = X + .5: NEXT
    Y:Y = 20: FOR X = 15 TO 28: PLOT X,Y:Y = Y - 1: NEXT X: GOTO 400
758 :
759 :
760  REM **--L--**
761  VLIN 5,35 AT 15: HLIN 15,25 AT 35: GOTO 400
768 :
769 :
770  REM **--M--**
771  VLIN 6,35 AT 11: VLIN 6,35 AT 30
772 X = 11:Y = 6
773  PLOT X,Y
774 X = X + 1:Y = Y + 1
775  IF X = 21 AND Y = 16 THEN 777
776  GOTO 773
777 X = 21: FOR Y = 15 TO 6 STEP  - 1: PLOT X,Y:X = X + 1: NEXT Y: GOTO
    400
778 :
779 :
780  REM **--N--**
781  VLIN 5,34 AT 14:X = 15: FOR Y = 5 TO 34: PLOT X,Y:X = X + .5: NEXT
    Y: VLIN 5,34 AT 30: GOTO 400
788 :
789 :
790  REM **--O--**
791  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,35 AT 25: HLIN 15,25 AT 35
    : GOTO 400
798 :
799 :
800  REM **--P--**
801  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,20 AT 25: HLIN 15,25 AT 20
    : GOTO 400
808 :
```

```
809 :
810  REM **--Q--**
811  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,35 AT 25: HLIN 15,25 AT 35
    : PLOT 26,36: PLOT 27,37: PLOT 28,38: PLOT 29,39: GOTO 400
818 :
819 :
820  REM **--R--**
821  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,15 AT 25: HLIN 15,25 AT 15
    :X = 18: FOR Y = 16 TO 35: PLOT X,Y:X = X + .5: NEXT Y: GOTO 400
828 :
829 :
830  REM **--S--**
831  HLIN 15,25 AT 5: VLIN 5,20 AT 15: HLIN 15,25 AT 20: VLIN 21,35 AT 2
    5: HLIN 15,25 AT 35: GOTO 400
838 :
839 :
840  REM **--T--**
841  HLIN 13,28 AT 5: VLIN 5,35 AT 20: GOTO 400
848 :
849 :
850  REM **--U--**
851  VLIN 5,35 AT 15: HLIN 15,25 AT 35: VLIN 5,35 AT 25: GOTO 400
858 :
859 :
860  REM **--V--**
861 X = 10: FOR Y = 5 TO 35: PLOT X,Y:X = X + .33: NEXT Y:X = 30: FOR Y =
    5 TO 35: PLOT X,Y:X = X - .33: NEXT Y: GOTO 400
868 :
869 :
870  REM **--W--**
871  VLIN 6,35 AT 10: VLIN 6,35 AT 30:X = 10: FOR Y = 35 TO 16 STEP  - 1
    : PLOT X,Y:X = X + .5: NEXT Y:X = 21: FOR Y = 16 TO 35: PLOT X,Y:X =
    X + .5: NEXT Y: PLOT 20,15: GOTO 400
878 :
879 :
880  REM **--X--**
881 X = 13: FOR Y = 5 TO 35: PLOT X,Y:X = X + .5: NEXT Y:X = 28: FOR Y =
    5 TO 35: PLOT X,Y:X = X - .5: NEXT Y: GOTO 400
888 :
889 :
890  REM **--Y--**
891  VLIN 19,39 AT 20:X = 8: FOR Y = 6 TO 18: PLOT X,Y:X = X + 1: NEXT Y
    :X = 32: FOR Y = 6 TO 18: PLOT X,Y:X = X - 1: NEXT Y: GOTO 400
898 :
899 :
900  REM **--Z--**
```

```
901  HLIN 15,30 AT 5: HLIN 15,30 AT 36:X = 30: FOR Y = 6 TO 35: PLOT X,Y
   :X = X - .5: NEXT Y: GOTO 400
998 :
999 :
1000  REM **--RETURN TO LETTERS MENU--**
1005  INVERSE
1010  VTAB 10: PRINT "THAT'S ALL FOLKS!!"
1015  NORMAL
1020  PRINT D$;"RUN LETTERS.MENU"
1998 :
1999 :
2000  DATA  A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,END
```

Type Following Letter

```
100  REM ***--TYPE.LETT.FOLL--***
110 :
120 :
130 D$ =  CHR$ (4): REM  CONTROL D
140  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
150 :
160 :
170  REM **--USER INSTRUCTIONS--**
180  HOME
190  INVERSE : HTAB 13: PRINT "INSTRUCTIONS": NORMAL
200  VTAB 6
210  PRINT "PRESS THE CORRECT KEY FOR"
220  PRINT
230  PRINT "THE LETTER THAT FOLLOWS THE"
240  PRINT
250  PRINT "LETTER DISPLAYED ON THE SCREEN."
260  VTAB 15
270  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT " KE
   Y TO END THE PROGRAM."
280  PRINT
285  PRINT : PRINT
290  PRINT "PRESS THE ";: INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT "
    TO BEGIN.";
300  GET L$: PRINT
310  IF  ASC (L$) <  > 32 THEN 260
320  GOTO 510: REM  DISPLAY LETTER ROUTINE
330 :
340 :
400  REM **--GET LETTER--**
```

```
410 LT = LT + 1
420  IF LT > 90 THEN LT = 65
430 LTR$ =  CHR$ (LT)
440  GET LR$
450  IF  ASC (LR$) = 13 THEN  TEXT : HOME : GOTO 1000
460  IF LR$ = LTR$ THEN 510
470  IF  ASC (LR$) > 96 AND  ASC (LR$) < 123 THEN LR$ =  CHR$ ( ASC (LR$
    ) - 32): GOTO 460: REM  TEST AGAIN
480  GOTO 440: REM  INVALID ENTRY TRY AGAIN
490 :
500 :
510  REM **--DISPLAY LETTER--**
520  GR
530 CLR =  INT ( RND (1) * 100)
540  IF CLR = 0 THEN 530
550  IF CLR > 15 THEN 530
560  COLOR= CLR
570 LT =  INT ( RND (1) * 100)
580  IF LT < 65 OR LT > 90 THEN 570
590 LTR$ =  CHR$ (LT)
598 :
599 :
600  REM **--GOTO DRAWING ROUTINE--**
601  IF LTR$ = "A" THEN 650
602  IF LTR$ = "B" THEN 660
603  IF LTR$ = "C" THEN 670
604  IF LTR$ = "D" THEN 680
605  IF LTR$ = "E" THEN 690
606  IF LTR$ = "F" THEN 700
607  IF LTR$ = "G" THEN 710
608  IF LTR$ = "H" THEN 720
609  IF LTR$ = "I" THEN 730
610  IF LTR$ = "J" THEN 740
611  IF LTR$ = "K" THEN 750
612  IF LTR$ = "L" THEN 760
613  IF LTR$ = "M" THEN 770
614  IF LTR$ = "N" THEN 780
615  IF LTR$ = "O" THEN 790
616  IF LTR$ = "P" THEN 800
617  IF LTR$ = "Q" THEN 810
618  IF LTR$ = "R" THEN 820
619  IF LTR$ = "S" THEN 830
620  IF LTR$ = "T" THEN 840
621  IF LTR$ = "U" THEN 850
622  IF LTR$ = "V" THEN 860
623  IF LTR$ = "W" THEN 870
```

```
624  IF LTR$ = "X" THEN 880
625  IF LTR$ = "Y" THEN 890
626  IF LTR$ = "Z" THEN 900
627 :
628 :
650  REM **--A--**
651 X = 20: FOR Y = 5 TO 35: PLOT X,Y:X = X - .33: NEXT Y:X = 20: FOR Y =
    5 TO 35: PLOT X,Y:X = X + .33: NEXT Y: HLIN 15,25 AT 20: PLOT 19,5: GOTO
    400
658 :
659 :
660  REM **--B--**
661  VLIN 5,35 AT 15: HLIN 13,25 AT 5: VLIN 5,20 AT 25: HLIN 17,25 AT 20
    : HLIN 17,25 AT 20: VLIN 21,35 AT 25: HLIN 13,25 AT 35: GOTO 400
668 :
669 :
670  REM **--C--**
671  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,25 AT 35: GOTO 400
678 :
679 :
680  REM **--D--**
681  VLIN 5,35 AT 15: HLIN 13,25 AT 5: VLIN 5,35 AT 25: HLIN 13,25 AT 35
    : GOTO 400
688 :
689 :
690  REM **--E--**
691  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,20 AT 20: HLIN 15,25 AT 3
    5: GOTO 400
698 :
699 :
700  REM **--F--**
701  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,20 AT 20: GOTO 400
708 :
709 :
710  REM **--G--**
711  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,25 AT 35: VLIN 25,35 AT 2
    5: HLIN 20,25 AT 25: GOTO 400
718 :
719 :
720  REM **--H--**
721  VLIN 5,35 AT 15: VLIN 5,35 AT 25: HLIN 15,25 AT 20: GOTO 400
728 :
729 :
730  REM **--I--**
731  VLIN 5,35 AT 20: HLIN 15,25 AT 5: HLIN 15,25 AT 35: GOTO 400
738 :
```

```
739 :
740  REM **--J--**
741  VLIN 5,35 AT 25: HLIN 15,25 AT 35: VLIN 30,35 AT 15: HLIN 20,30 AT
    5: GOTO 400
748 :
749 :
750  REM **--K--**
751  VLIN 5,35 AT 15:X = 20: FOR Y = 15 TO 35: PLOT X,Y:X = X + .5: NEXT
    Y:Y = 20: FOR X = 15 TO 28: PLOT X,Y:Y = Y - 1: NEXT X: GOTO 400
758 :
759 :
760  REM **--L--**
761  VLIN 5,35 AT 15: HLIN 15,25 AT 35: GOTO 400
768 :
769 :
770  REM **--M--**
771  VLIN 6,35 AT 11: VLIN 6,35 AT 30
772 X = 11:Y = 6
773  PLOT X,Y
774 X = X + 1:Y = Y + 1
775  IF X = 21 AND Y = 16 THEN 777
776  GOTO 773
777 X = 21: FOR Y = 15 TO 6 STEP  - 1: PLOT X,Y:X = X + 1: NEXT Y: GOTO
    400
778 :
779 :
780  REM **--N--**
781  VLIN 5,34 AT 14:X = 15: FOR Y = 5 TO 34: PLOT X,Y:X = X + .5: NEXT
    Y: VLIN 5,34 AT 30: GOTO 400
788 :
789 :
790  REM **--O--**
791  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,35 AT 25: HLIN 15,25 AT 35
    : GOTO 400
798 :
799 :
800  REM **--P--**
801  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,20 AT 25: HLIN 15,25 AT 20
    : GOTO 400
808 :
809 :
810  REM **--Q--**
811  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,35 AT 25: HLIN 15,25 AT 35
    : PLOT 26,36: PLOT 27,37: PLOT 28,38: PLOT 29,39: GOTO 400
818 :
819 :
```

```
820  REM **--R--**
821  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,15 AT 25: HLIN 15,25 AT 15
    :X = 18: FOR Y = 16 TO 35: PLOT X,Y:X = X + .5: NEXT Y: GOTO 400
828 :
829 :
830  REM **--S--**
831  HLIN 15,25 AT 5: VLIN 5,20 AT 15: HLIN 15,25 AT 20: VLIN 21,35 AT 2
    5: HLIN 15,25 AT 35: GOTO 400
838 :
839 :
840  REM **--T--**
841  HLIN 13,28 AT 5: VLIN 5,35 AT 20: GOTO 400
848 :
849 :
850  REM **--U--**
851  VLIN 5,35 AT 15: HLIN 15,25 AT 35: VLIN 5,35 AT 25: GOTO 400
858 :
859 :
860  REM **--V--**
861 X = 10: FOR Y = 5 TO 35: PLOT X,Y:X = X + .33: NEXT Y:X = 30: FOR Y =
    5 TO 35: PLOT X,Y:X = X - .33: NEXT Y: GOTO 400
868 :
869 :
870  REM **--W--**
871  VLIN 6,35 AT 10: VLIN 6,35 AT 30:X = 10: FOR Y = 35 TO 16 STEP  - 1
    : PLOT X,Y:X = X + .5: NEXT Y:X = 21: FOR Y = 16 TO 35: PLOT X,Y:X =
    X + .5: NEXT Y: PLOT 20,15: GOTO 400
878 :
879 :
880  REM **--X--**
881 X = 13: FOR Y = 5 TO 35: PLOT X,Y:X = X + .5: NEXT Y:X = 28: FOR Y =
    5 TO 35: PLOT X,Y:X = X - .5: NEXT Y: GOTO 400
888 :
889 :
890  REM **--Y--**
891  VLIN 19,39 AT 20:X = 8: FOR Y = 6 TO 18: PLOT X,Y:X = X + 1: NEXT Y
    :X = 32: FOR Y = 6 TO 18: PLOT X,Y:X = X - 1: NEXT Y: GOTO 400
898 :
899 :
900  REM **--Z--**
901  HLIN 15,30 AT 5: HLIN 15,30 AT 36:X = 30: FOR Y = 6 TO 35: PLOT X,Y
    :X = X - .5: NEXT Y: GOTO 400
998 :
999 :
1000  REM **--RETURN TO LETTERS MENU--**
1010  PRINT
1020  PRINT D$;"RUN LETTERS.MENU"
```

Type Preceding Letter

```
100  REM  ***--TYPE.LET.BEFORE--***
110 :
120 :
130 D$ =  CHR$ (4): REM  CONTROL D
140  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
150 :
160 :
170  REM **--USER INSTRUCTIONS--**
180  HOME
190  INVERSE : HTAB 13: PRINT "INSTRUCTIONS": NORMAL
200  VTAB 6
210  PRINT "PRESS THE CORRECT KEY FOR"
220  PRINT
230  PRINT "THE LETTER THAT COMES BEFORE THE"
240  PRINT
250  PRINT "LETTER DISPLAYED ON THE SCREEN."
260  VTAB 15
270  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT " KE
    Y TO END THE PROGRAM."
280  PRINT
285  PRINT : PRINT
290  PRINT "PRESS THE ";: INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT "
     TO BEGIN.";
300  GET L$: PRINT
310  IF  ASC (L$) <  > 32 THEN 260
320  GOTO 510: REM  DISPLAY LETTER ROUTINE
330 :
340 :
400  REM **--GET LETTER--**
410 LT = LT - 1
420  IF LT < 65 THEN LT = 90
430 LTR$ =  CHR$ (LT)
440  GET LR$
450  IF  ASC (LR$) = 13 THEN  TEXT : HOME : GOTO 1000
460  IF LR$ = LTR$ THEN 510
470  IF  ASC (LR$) > 96 AND  ASC (LR$) < 123 THEN LR$ =  CHR$ ( ASC (LR$
    ) - 32): GOTO 460: REM  TEST AGAIN
480  GOTO 440: REM  INVALID ENTRY TRY AGAIN
490 :
500 :
510  REM **--DISPLAY LETTER--**
520  GR
530 CLR =  INT ( RND (1) * 100)
540  IF CLR = 0 THEN 530
550  IF CLR > 15 THEN 530
```

```
560  COLOR= CLR
570 LT =  INT ( RND (1) * 100)
580  IF LT < 65 OR LT > 90 THEN 570
590 LTR$ =  CHR$ (LT)
598 :
599 :
600  REM **--GOTO DRAWING ROUTINE--**
601  IF LTR$ = "A" THEN 650
602  IF LTR$ = "B" THEN 660
603  IF LTR$ = "C" THEN 670
604  IF LTR$ = "D" THEN 680
605  IF LTR$ = "E" THEN 690
606  IF LTR$ = "F" THEN 700
607  IF LTR$ = "G" THEN 710
608  IF LTR$ = "H" THEN 720
609  IF LTR$ = "I" THEN 730
610  IF LTR$ = "J" THEN 740
611  IF LTR$ = "K" THEN 750
612  IF LTR$ = "L" THEN 760
613  IF LTR$ = "M" THEN 770
614  IF LTR$ = "N" THEN 780
615  IF LTR$ = "O" THEN 790
616  IF LTR$ = "P" THEN 800
617  IF LTR$ = "Q" THEN 810
618  IF LTR$ = "R" THEN 820
619  IF LTR$ = "S" THEN 830
620  IF LTR$ = "T" THEN 840
621  IF LTR$ = "U" THEN 850
622  IF LTR$ = "V" THEN 860
623  IF LTR$ = "W" THEN 870
624  IF LTR$ = "X" THEN 880
625  IF LTR$ = "Y" THEN 890
626  IF LTR$ = "Z" THEN 900
627 :
628 :
650  REM **--A--**
651 X = 20: FOR Y = 5 TO 35: PLOT X,Y:X = X - .33: NEXT Y:X = 20: FOR Y =
    5 TO 35: PLOT X,Y:X = X + .33: NEXT Y: HLIN 15,25 AT 20: PLOT 19,5: GOTO
    400
658 :
659 :
660  REM **--B--**
661  VLIN 5,35 AT 15: HLIN 13,25 AT 5: VLIN 5,20 AT 25: HLIN 17,25 AT 20
   : HLIN 17,25 AT 20: VLIN 21,35 AT 25: HLIN 13,25 AT 35: GOTO 400
668 :
669 :
```

```
670  REM **--C--**
671  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,25 AT 35: GOTO 400
678 :
679 :
680  REM **--D--**
681  VLIN 5,35 AT 15: HLIN 13,25 AT 5: VLIN 5,35 AT 25: HLIN 13,25 AT 35
    : GOTO 400
688 :
689 :
690  REM **--E--**
691  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,20 AT 20: HLIN 15,25 AT 3
    5: GOTO 400
698 :
699 :
700  REM **--F--**
701  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,20 AT 20: GOTO 400
708 :
709 :
710  REM **--G--**
711  VLIN 5,35 AT 15: HLIN 15,25 AT 5: HLIN 15,25 AT 35: VLIN 25,35 AT 2
    5: HLIN 20,25 AT 25: GOTO 400
718 :
719 :
720  REM **--H--**
721  VLIN 5,35 AT 15: VLIN 5,35 AT 25: HLIN 15,25 AT 20: GOTO 400
728 :
729 :
730  REM **--I--**
731  VLIN 5,35 AT 20: HLIN 15,25 AT 5: HLIN 15,25 AT 35: GOTO 400
738 :
739 :
740  REM **--J--**
741  VLIN 5,35 AT 25: HLIN 15,25 AT 35: VLIN 30,35 AT 15: HLIN 20,30 AT
    5: GOTO 400
748 :
749 :
750  REM **--K--**
751  VLIN 5,35 AT 15:X = 20: FOR Y = 15 TO 35: PLOT X,Y:X = X + .5: NEXT
    Y:Y = 20: FOR X = 15 TO 28: PLOT X,Y:Y = Y - 1: NEXT X: GOTO 400
758 :
759 :
760  REM **--L--**
761  VLIN 5,35 AT 15: HLIN 15,25 AT 35: GOTO 400
768 :
769 :
770  REM **--M--**
```

```
771  VLIN 6,35 AT 11: VLIN 6,35 AT 30
772 X = 11:Y = 6
773  PLOT X,Y
774 X = X + 1:Y = Y + 1
775  IF X = 21 AND Y = 16 THEN 777
776  GOTO 773
777 X = 21: FOR Y = 15 TO 6 STEP  - 1: PLOT X,Y:X = X + 1: NEXT Y: GOTO
    400
778 :
779 :
780  REM **--N--**
781  VLIN 5,34 AT 14:X = 15: FOR Y = 5 TO 34: PLOT X,Y:X =X + .5: NEXT
    Y: VLIN 5,34 AT 30: GOTO 400
788 :
789 :
790  REM **--O--**
791  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,35 AT 25: HLIN 15,25 AT 35
    : GOTO 400
798 :
799 :
800  REM **--P--**
801  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,20 AT 25: HLIN 15,25 AT 20
    : GOTO 400
808 :
809 :
810  REM **--Q--**
811  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,35 AT 25: HLIN 15,25 AT 35
    : PLOT 26,36: PLOT 27,37: PLOT 28,38: PLOT 29,39: GOTO 400
818 :
819 :
820  REM **--R--**
821  VLIN 5,35 AT 15: HLIN 15,25 AT 5: VLIN 5,15 AT 25: HLIN 15,25 AT 15
    :X = 18: FOR Y = 16 TO 35: PLOT X,Y:X = X + .5: NEXT Y: GOTO 400
828 :
829 :
830  REM **--S--**
831  HLIN 15,25 AT 5: VLIN 5,20 AT 15: HLIN 15,25 AT 20: VLIN 21,35 AT 2
    5: HLIN 15,25 AT 35: GOTO 400
838 :
839 :
840  REM **--T--**
841  HLIN 13,28 AT 5: VLIN 5,35 AT 20: GOTO 400
848 :
849 :
850  REM **--U--**
851  VLIN 5,35 AT 15: HLIN 15,25 AT 35: VLIN 5,35 AT 25: GOTO 400
```

```
858 :
859 :
860  REM **--V--**
861 X = 10: FOR Y = 5 TO 35: PLOT X,Y:X = X + .33: NEXT Y:X = 30: FOR Y =
    5 TO 35: PLOT X,Y:X = X - .33: NEXT Y: GOTO 400
868 :
869 :
870  REM **--W--**
871  VLIN 6,35 AT 10: VLIN 6,35 AT 30:X = 10: FOR Y = 35 TO 16 STEP  - 1
    : PLOT X,Y:X = X + .5: NEXT Y:X = 21: FOR Y = 16 TO 35: PLOT X,Y:X =
    X + .5: NEXT Y: PLOT 20,15: GOTO 400
878 :
879 :
880  REM **--X--**
881 X = 13: FOR Y = 5 TO 35: PLOT X,Y:X = X + .5: NEXT Y:X = 28: FOR Y =
    5 TO 35: PLOT X,Y:X = X - .5: NEXT Y: GOTO 400
888 :
889 :
890  REM **--Y--**
891  VLIN 19,39 AT 20:X = 8: FOR Y = 6 TO 18: PLOT X,Y:X = X + 1: NEXT Y
    :X = 32: FOR Y = 6 TO 18: PLOT X,Y:X = X - 1: NEXT Y: GOTO 400
898 :
899 :
900  REM **--Z--**
901  HLIN 15,30 AT 5: HLIN 15,30 AT 36:X = 30: FOR Y = 6 TO 35: PLOT X,Y
    :X = X - .5: NEXT Y: GOTO 400
998 :
999 :
1000  REM **--RETURN TO LETTERS MENU--**
1010  PRINT
1020  PRINT D$;"RUN LETTERS.MENU"
```

Moving Letter I

```
100  REM  ***--MOVING.I--***
110 :
120 :
130  REM  **--INITIALIZE LOW-RES GRAPHICS--**
140  GR
150  COLOR= 6: REM  SET COLOR TO BLUE
160 VERTLGTH = 30:HZLGTH =  INT (VERTLGTH / 6):X = 20:Y = 5
170  GOSUB 500: INPUT "WHICH METHOD (K/P) ";METHOD$: COLOR= 0: GOSUB 500

180 KEY =  PEEK ( - 16384)
185 SA =  PEEK ( - 16286): REM STORE VALUE OF BUTTON 1/SOLID APPLE KEY
```

```
187 OA =  PEEK ( - 16287): REM STORE VALUE OF BUTTON 0/OPEN APPLE KEY
189  IF METHOD$ = "P" OR METHOD$ = "p" OR METHOD$ = "J" OR METHOD$ = "j"
     THEN 2000: REM    SKIP OVER KEYBOARD LINES
190  IF (KEY - 128) = 8 THEN  COLOR= 0: GOSUB 500:X = X - 1: GOTO 300: REM
       LEFT ARROW PRESSED
200  IF (KEY - 128) = 21 THEN  COLOR= 0: GOSUB 500:X = X + 1: GOTO 300: REM
      RIGHT ARROW PRESSED
210  IF (KEY - 128) = 10 THEN  COLOR= 0: GOSUB 500:Y = Y + 1: GOTO 300: REM
      DOWN ARROW PRESSED
220  IF (KEY - 128) = 11 THEN  COLOR= 0: GOSUB 500:Y = Y - 1: GOTO 300: REM
       UP ARROW PRESSED
225  IF SA > 127 THEN  COLOR= 0: GOSUB 500:VERTLGTH = VERTLGTH - 1:HZLGT
    H =  INT (VERTLGTH / 6): GOTO 300: REM  SOLID APPLE PRESSED
227  IF OA > 127 THEN  COLOR= 0: GOSUB 500:VERTLGTH = VERTLGTH + 1:HZLGT
    H =  INT (VERTLGTH / 6): GOTO 300: REM  OPEN APPLE PRESSED
230  IF (KEY - 128) = 27 THEN 400: REM  ESC KEY PRESSED SO END
240  GOTO 180
300  COLOR= 2: GOSUB 500
320  POKE  - 16368,0
330  GOTO 180
380 :
390 :
400  REM  **--END PROGRAM--**
405  POKE  - 16368,0
410  TEXT : HOME : END
440 :
450 :
460 :
500  REM  **--DRAW CAPITAL LETTER I--**
510  IF X - HZLGTH < 0 THEN X = HZLGTH: GOTO 530
520  IF X + HZLGTH > 39 THEN X = 39 - HZLGTH
530  IF Y + VERTLGTH > 39 THEN Y = 39 - VERTLGTH
540  IF Y < 0 THEN Y = 0
550  IF VERTLGTH > 39 THEN VERTLGTH = 39
560  IF VERTLGTH < 0 THEN VERTLGTH = 0
570  VLIN Y,Y + VERTLGTH AT X
580  HLIN X - HZLGTH,X + HZLGTH AT Y
590  HLIN X - HZLGTH,X + HZLGTH AT Y + VERTLGTH
600  RETURN
610 :
620 :
2000  REM  **--PADDLE POSITION--**
2010 PO =  PDL (0)
2020 :
2030 P1 =  PDL (1)
2040 :
2050  IF P1 > 127 AND P1 < 132 THEN 2080
2060  IF P1 < 131 THEN  COLOR= 0: GOSUB 500:Y = Y - 1
```

```
2070  IF P1 > 131 THEN  COLOR= 0: GOSUB 500:Y = Y + 1
2080  IF P0 > 127 AND P0 < 136 THEN 3000
2090  IF P0 > 136 THEN  COLOR= 0: GOSUB 500:X = X + 1
2100  IF P0 < 136 THEN  COLOR= 0: GOSUB 500:X = X - 1
2110  IF Y < 0 THEN Y = 0
2120 :
2130 :
3000  REM  **--APPLE KEYS--**
3010  IF SA > 127 THEN  COLOR= 0: GOSUB 500:VERTLGTH = VERTLGTH - 1:HZLG
    TH =  INT (VERTLGTH / 6): GOTO 300: REM    SOLID APPLE PRESSED
3020  IF OA > 127 THEN  COLOR= 0: GOSUB 500:VERTLGTH = VERTLGTH + 1:HZLG
    TH =  INT (VERTLGTH / 6): GOTO 300: REM   OPEN APPLE PRESSED
3030  IF (KEY - 128) = 27 THEN 400: REM   ESC KEY PRESSED SO END
3040  GOTO 300
```

Game Pattern Menu

```
100  REM  ***--GAME.PATT.MENU--***
110 :
120 :
130 TB = 8
140 D$ =  CHR$ (4): REM CONTROL D
150 :
160 :
170  REM **--DISPLAY MENU--**
180  TEXT : HOME
190  VTAB 1
200  HTAB TB + 4
210  INVERSE
220  PRINT "GAME/PATTERN MENU"
230  NORMAL
240  PRINT
250  HTAB TB
260  PRINT "1.  HOUSE PICTURE"
270  PRINT : HTAB TB
280  PRINT "2.  LOW-RES PATTERN"
290  PRINT : HTAB TB
300  PRINT "3.  WALL BLASTER"
310  PRINT : HTAB TB
320  PRINT "4.  SWEEPER"
330  PRINT : HTAB TB
340  PRINT "5.  MAZE"
350  PRINT : HTAB TB
360  PRINT "6.  FREE DRAWING"
370  PRINT : HTAB TB
380  PRINT "7.  DISPLAY DRAWING"
```

```
390  PRINT : HTAB TB
400  PRINT "8.  CATALOG"
410  PRINT : HTAB TB
415  PRINT "9.  END"
419  PRINT : HTAB TB
420  PRINT "WHICH NUMBER PLEASE? (";: INVERSE : PRINT "1-9";: NORMAL : PRINT
    "):";
430  GET NB$: PRINT
440 NB =  VAL (NB$)
450  IF NB < 1 OR NB > 9 THEN  PRINT : HTAB TB: INVERSE : PRINT "INCORRE
    CT CHOICE!": NORMAL : GOTO 190
460  IF NB = 1 THEN 1000
470  IF NB = 2 THEN 2000
480  IF NB = 3 THEN 3000
490  IF NB = 4 THEN 4000
500  IF NB = 5 THEN 5000
510  IF NB = 6 THEN 6000
520  IF NB = 7 THEN 7000
530  IF NB = 8 THEN 8000
535  IF NB = 9 THEN 9000
540 :
550 :
1000  REM **--HOUSE PICTURE--**
1010  HOME
1020  PRINT "This selection will show a very simple"
1025  PRINT
1030  PRINT "drawing of a house. The basic design"
1031  PRINT
1032  PRINT "was done by a student after 1 brief"
1033  PRINT
1034  PRINT "lesson. I have added the color and"
1035  PRINT
1036  PRINT "a few details. The program has been"
1037  PRINT
1038  PRINT "intentionally left in its primitive"
1039  PRINT
1040  PRINT "state to show what can be done with-"
1041  PRINT
1042  PRINT "out elaborate technique. When you are"
1043  PRINT
1044  PRINT "finished viewing the picture, press"
1045  PRINT
1050  PRINT "the 'RETURN' key to go back to the menu."
1055  VTAB 22
1060  PRINT "PRESS THE ";
1065  INVERSE
1070  PRINT "return";
1075  NORMAL
```

```
1080  INPUT " KEY TO CONTINUE: ";L$
1100  PRINT D$;"RUN HOUSE.PICTURE"
1980 :
1990 :
2000  REM **--LOW-RES. PATTERN--**
2010  HOME
2030  PRINT "This option alternates between patterns"
2032  PRINT
2034  PRINT "1 and 2. No. 1 displays a pattern that"
2040  PRINT
2050  PRINT "constantly changes. It will cycle"
2060  PRINT
2070  PRINT "through all the low-resolution colors."
2080  PRINT
2090  PRINT "When you want to stop the pattern,press"
2100  PRINT
2110  PRINT "the 'RETURN' key and you will go back"
2120  PRINT
2130  PRINT "to the menu. The next time you choose"
2140  PRINT
2150  PRINT "this option, you will see pattern #2."
2175  PRINT
2180  PRINT "Pattern 2 displays a form of animation"
2190  PRINT
2200  PRINT "with two bouncing bricks exchanging"
2210  PRINT
2220  PRINT "the colors in two diagonal corners."
2230  GOSUB 9500: REM  RETURN KEY ROUTINE
2240  HOME
2250  PRINT "This pattern also will continue until"
2255  PRINT
2260  PRINT "you press the 'RETURN' key. You will "
2265  PRINT
2270  PRINT "then go back to the menu and pattern 1"
2275  PRINT
2280  PRINT "will be available."
2700  GOSUB 9500: REM  RETURN KEY ROUTINE
2702  HOME : VTAB 10: INVERSE
2703  PRINT "ONE MOMENT PLEASE!": NORMAL
2705  PRINT D$;"RENAME LOWRES.PATTRN.1,TEMP"
2710  PRINT D$;"RENAME LOWRES.PATTRN.2,LOWRES.PATTRN.1"
2720  PRINT D$;"RENAME TEMP,LOWRES.PATTRN.2"
2800  PRINT D$;"RUN LOWRES.PATTRN.1"
2980 :
2990 :
3000  REM **--WALL BLASTER--**
3010  PRINT D$;"RUN WALL.BLASTER"
3980 :
```

```
3990 :
4000  REM **--SWEEPER--**
4010  PRINT D$;"RUN SWEEPER"
4980 :
4990 :
5000  REM **--MAZE--**
5010  PRINT D$;"RUN MAZE"
5980 :
5990 :
6000  REM **--FREE DRAWING--**
6010  PRINT D$;"RUN FREE.DRAWING"
6980 :
6990 :
7000  REM **--DISPLAY DRAWING--**
7010  PRINT D$;"RUN VIEW.DRAWING"
7980 :
7990 :
8000  REM **--DISPLAY CATALOG--**
8010  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
8020  HOME
8030  PRINT D$;"CATALOG"
8040  PRINT
8050  POKE  - 16368,0: REM CLEAR KBRD BUFFER
8060  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT " K
    EY TO CONTINUE:";L$
8070  GOTO 170
8080 :
8090 :
9000  REM **--END--**
9010  HOME
9020  VTAB 10
9030  INVERSE
9040  PRINT "SEE YOU NEXT TIME."
9050  NORMAL
9060  END
9070 :
9080 :
9090 :
9100 :
9500  REM **--RETURN KEY ROUTINE--**
9510  VTAB 23
9520  PRINT "PRESS THE ";
9530  INVERSE
9540  PRINT "RETURN";
9550  NORMAL
9560  INPUT " KEY TO CONTINUE: ";L$
9570  RETURN
9996 :
```

```
9997 :
9998 :
9999 :
10000  REM  GAME/PATTERN SYSTEM
11000  REM  COPYRIGHT OCTOBER 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS RD
14000  REM  CORVALLIS OR. 97330-9340
```

House Picture

```
10  GR
15  POKE  - 16302,0: REM  ADDED TO SHOW FULL SCREEN
17  CALL  - 1998: REM  ADDED TO FIRST CLEAR SCREEN
20  COLOR= 1
30  PLOT 20,0
40  PLOT 19,1
50  PLOT 21,1
60  PLOT 18,2
70  PLOT 22,2
80  PLOT 17,3
90  PLOT 23,3
100  PLOT 16,4
110  PLOT 24,4
120  PLOT 15,5
130  PLOT 25,5
140  PLOT 14,6
150  PLOT 26,6
160  PLOT 13,7
170  PLOT 27,7
180  PLOT 12,8
190  PLOT 28,8
197  COLOR= 13
200  VLIN 8,39 AT 12
210  VLIN 8,39 AT 28
220  HLIN 12,28 AT 8
230  HLIN 12,28 AT 23
240  HLIN 12,28 AT 39
247  COLOR= 15
250  VLIN 30,39 AT 18
260  VLIN 30,39 AT 19
270  VLIN 30,39 AT 20
280  VLIN 30,39 AT 21
290  VLIN 25,28 AT 14
300  VLIN 25,28 AT 15
310  VLIN 25,28 AT 16
320  VLIN 25,28 AT 26
```

```
330  VLIN 25,28 AT 24
340  VLIN 25,28 AT 25
350  VLIN 10,13 AT 14
360  VLIN 10,13 AT 15
370  VLIN 10,13 AT 16
380  VLIN 10,13 AT 24
390  VLIN 10,13 AT 25
400  VLIN 10,13 AT 26
405  COLOR= 8
410  VLIN 15,39 AT 5
420  VLIN 15,39 AT 6
425  COLOR= 4
430  PLOT 3,15
440  PLOT 3,16
450  PLOT 2,17
460  PLOT 2,18
470  PLOT 2,19
480  PLOT 2,20
490  PLOT 4,15
500  PLOT 7,15
510  PLOT 8,15
520  PLOT 9,16
530  PLOT 9,17
540  PLOT 9,18
550  PLOT 9,19
560  PLOT 9,20
570  PLOT 33,21
580  PLOT 32,22
590  PLOT 33,22
600  PLOT 34,22
610  PLOT 31,23
620  PLOT 32,23
630  PLOT 33,23
640  PLOT 34,23
650  PLOT 35,23
660  PLOT 31,24
670  PLOT 32,24
680  PLOT 33,24
690  PLOT 34,24
700  PLOT 35,24
710  PLOT 30,25
720  PLOT 31,25
730  PLOT 32,25
740  PLOT 33,25
750  PLOT 34,25
760  PLOT 35,25
770  PLOT 36,25
780  PLOT 29,26
```

```
790  PLOT 30,26
800  PLOT 31,26
810  PLOT 32,26
820  PLOT 33,26
830  PLOT 34,26
840  PLOT 35,26
850  PLOT 36,26
860  PLOT 37,26
865  COLOR= 8
870  VLIN 27,39 AT 33
880  COLOR= 5
900  VLIN 0,4 AT 16
910  VLIN 0,5 AT 15
915  COLOR= 0
917  PLOT 21,35
920  COLOR= 14
930  HLIN 18,21 AT 40
940  HLIN 18,21 AT 41
950  COLOR= 6
960  HLIN 0,14 AT 0
970  HLIN 0,14 AT 1
980  HLIN 0,14 AT 2
990  HLIN 0,14 AT 3
1000  HLIN 0,14 AT 4
1010  HLIN 0,14 AT 5
1011  HLIN 0,13 AT 6
1012  HLIN 0,12 AT 7
1020  PLOT 17,0
1025  PLOT 18,0
1030  PLOT 19,0
1035  PLOT 17,1
1040  PLOT 18,1
1045  PLOT 17,2
1050  HLIN 21,39 AT 0
1055  HLIN 22,39 AT 1
1060  HLIN 23,39 AT 2
1065  HLIN 24,39 AT 3
1070  HLIN 25,39 AT 4
1075  HLIN 26,39 AT 5
1080  HLIN 27,39 AT 6
1085  HLIN 28,39 AT 7
1090  HLIN 29,39 AT 8
1095  HLIN 0,11 AT 8
1100  HLIN 0,11 AT 9
1110  HLIN 0,11 AT 10
1120  HLIN 0,11 AT 11
1130  HLIN 0,11 AT 12
1140  HLIN 0,11 AT 13
```

```
1150  HLIN 0,11 AT 14
1155  HLIN 29,39 AT 9
1160  HLIN 29,39 AT 10
1170  HLIN 29,39 AT 11
1180  HLIN 29,39 AT 12
1190  HLIN 29,39 AT 13
1191  HLIN 29,39 AT 14
1192  HLIN 29,39 AT 15
1193  HLIN 29,39 AT 16
1194  HLIN 29,39 AT 17
1195  HLIN 29,39 AT 18
1196  HLIN 29,39 AT 19
1197  HLIN 29,39 AT 20
1200  HLIN 0,2 AT 15
1210  HLIN 0,2 AT 16
1220  HLIN 9,11 AT 15
1230  HLIN 10,11 AT 16
1240  HLIN 10,11 AT 17
1250  HLIN 10,11 AT 18
1260  HLIN 10,11 AT 19
1265  HLIN 10,11 AT 20
1270  HLIN 0,1 AT 17
1280  HLIN 0,1 AT 18
1285  HLIN 0,1 AT 19
1290  HLIN 0,1 AT 20
1295  HLIN 0,4 AT 21
1300  HLIN 7,11 AT 21
1305  PLOT 4,16
1310  PLOT 7,16
1315  PLOT 8,16
1320  HLIN 7,8 AT 17
1325  HLIN 7,8 AT 18
1330  HLIN 7,8 AT 19
1335  HLIN 7,8 AT 20
1340  HLIN 3,4 AT 17
1345  HLIN 3,4 AT 18
1350  HLIN 3,4 AT 19
1355  HLIN 3,4 AT 20
1480 :
1490 :
1500  REM  NOTICE THAT WITHOUT REM STATEMENTS IT IS
1510  REM  VERY DIFFICULT TO KNOW WHAT PART OF THE
1520  REM  PICTURE EACH INSTRUCTION IS DRAWING. STILL
1530  REM  THE RESULTS ARE QUITE GOOD FOR 1 LESSON.
1540  REM  HINT--THE ADDED COLOR STATEMENTS GIVE SOME
1550  REM  INDICATION OF WHERE DIFFERENT PARTS START.
1880 :
1890 :
```

```
1900  REM  **--DISPLAY PICTURE UNTIL KEY IS PRESSED--**
1910 KEY =  PEEK ( - 16384)
1920  IF KEY < 128 THEN 1900
1930  TEXT
1935  POKE  - 16368,0
1940  PRINT  CHR$ (4);"RUN GAME.PATT.MENU"
1950  GOTO 1900
```

Low-Res Pattern 1

```
100  REM  ***--LOWRES.PATTRN.1--***
110 :
120 :
130  REM X IS HORIZONTAL POSITION
140  REM Y IS VERTICAL POSITION
150  REM B IS HORIZONTAL BEG. POINT
160  REM C IS COLOR INCREMENT
170  REM Z IS FLAG
180 :
190 :
200  HOME
210  GR :B = 0:Y = 0
220  POKE  - 16368,0: REM CLEAR KBRD BUFFER
230 :
240 :
250  REM **--RETURN TO MENU ROUTINE--**
260 KEY =  PEEK ( - 16384): REM CHECK KBRD FOR PRESSED KEY
270  IF KEY < 128 THEN 1000: REM < 128 = NO PRESSED KEY
280  TEXT
290  POKE  - 16368,0: REM CLEAR KBRD BUFFER
300  PRINT  CHR$ (4);"RUN GAME.PATT.MENU"
310 :
320 :
1000  REM **--UPPER LEFT QUADRANT--**
1010 :
1020  REM LOOP PLOTS 1 ROW
1030  FOR X = B TO 17 STEP 2
1040  COLOR= 13 + C
1050  PLOT X,Y
1060  NEXT X
1070 :
1080  REM NEXT LINE TESTS VERTICAL BOUNDARY
1090  IF Y < 19 THEN 1150
1100 :
1110  GOTO 2000: REM FINISHED WITH UPPER LEFT
1120 :
1130  REM B CHANGES AFTER EACH CYCLE
```

```
1140  REM INCREMENT THEN GO BACK FOR ANOTHER ROW
1150  IF B = 0 THEN B = 1:Y = Y + 1: GOTO 1030
1160  IF B = 1 THEN B = 0:Y = Y + 1: GOTO 1030
1170 :
1180 :
2000  REM **--LOWER RIGHT QUADRANT--**
2010 B = 21 + Z:Y = 20
2020  FOR X = B TO 39 STEP 2
2030  COLOR= 2 + C
2040  PLOT X,Y
2050  NEXT X
2060  IF Y < 39 THEN 2080
2070  GOTO 3000: REM FINISHED WITH LOWER RIGHT
2080  IF B = 22 THEN B = 21:Y = Y + 1: GOTO 2020
2090  IF B = 21 THEN B = 22:Y = Y + 1: GOTO 2020
2100 :
2110 :
3000  REM **--LOWER LEFT QUADRANT--**
3010 B = 0 + Z:Y = 20
3020  FOR X = B TO 17 STEP 2
3030  COLOR= 1 + C
3040  PLOT X,Y
3050  NEXT X
3060  IF Y < 39 THEN 3080
3070  GOTO 4000: REM FINISHED WITH LOWER LEFT
3080  IF B = 0 THEN B = 1:Y = Y + 1: GOTO 3020
3090  IF B = 1 THEN B = 0:Y = Y + 1: GOTO 3020
3100 :
3110 :
4000  REM **--UPPER RIGHT QUADRANT--**
4010 B = 21 + Z:Y = 0
4020  FOR X = B TO 39 STEP 2
4030  COLOR= 12 + C
4040  PLOT X,Y
4050  NEXT X
4060  IF Y < 19 THEN 4080
4070  GOTO 5000: REM FINISHED WITH UPPER RIGHT
4080  IF B = 22 THEN B = 21:Y = Y + 1: GOTO 4020
4090  IF B = 21 THEN B = 22:Y = Y + 1: GOTO 4020
4100 :
4110 :
5000  REM **--CENTER POLE--**
5010 B = 18 + Z:Y = 0
5020  FOR X = B TO 20 STEP 2
5030  COLOR= 15 + C
5040  PLOT X,Y
5050  NEXT X
5060  IF Y < 39 THEN 5080
```

```
5070  GOTO 6000: REM ALL FINISHED; BEGIN AGAIN
5080  IF B = 18 THEN B = 19:Y = Y + 1: GOTO 5020
5090  IF B = 19 THEN B = 18:Y = Y + 1: GOTO 5020
5100 :
5110 :
6000  REM **--RESET HORIZ. & VERT. POSITIONS--**
6010  IF Z = 1 THEN 6030
6020 B = 1:Y = 0:Z = 1: GOTO 250: REM BEGIN AGAIN
6030 B = 0:Y = 0:Z = 0:C = C + 1: GOTO 250: REM CHANGE COLOR; START OVER
```

Low-Res Pattern 2

```
100  REM ***--LOWRES.PATTRN.2--***
110 :
120 :
130  REM X IS HORIZONTAL POSITION
140  REM Y IS VERTICAL POSITION
150  REM B IS HORIZONTAL BEG. POINT
160  REM C IS COLOR INCREMENT
170  REM Z IS FLAG
180 :
190 :
200  HOME
210  GR :B = 0:Y = 0
220  POKE  - 16368,0: REM CLEAR KBRD BUFFER
230 :
240 :
250  REM **--RETURN TO MENU ROUTINE--**
260 KEY =  PEEK ( - 16384): REM CHECK KBRD FOR PRESSED KEY
270  IF KEY < 128 THEN 1000: REM < 128 = NO PRESSED KEY
280  TEXT
290  POKE  - 16368,0: REM CLEAR KBRD BUFFER
300  PRINT  CHR$ (4);"RUN GAME.PATT.MENU"
310 :
320 :
1000  REM **--UPPER LEFT QUADRANT--**
1010 :
1020  REM LOOP PLOTS 1 ROW
1030  FOR X = B TO 17 STEP 2
1040  COLOR= 13 + C
1050  PLOT X,Y
1060  NEXT X
1070 :
1080  REM NEXT LINE TESTS VERTICAL BOUNDARY
1090  IF Y < 19 THEN 1150
1100 :
```

```
1110  GOTO 2000: REM FINISHED WITH UPPER LEFT
1120 :
1130  REM B CHANGES AFTER EACH CYCLE
1140  REM INCREMENT THEN GO BACK FOR ANOTHER ROW
1150  IF B = 0 THEN B = 1:Y = Y + 1: GOTO 1030
1160  IF B = 1 THEN B = 0:Y = Y + 1: GOTO 1030
1170 :
1180 :
2000  REM **--LOWER RIGHT QUADRANT--**
2010 B = 22 + Z:Y = 20
2020  FOR X = B TO 39 STEP 2
2030  COLOR= 2 + C
2040  PLOT X,Y
2050  NEXT X
2060  IF Y < 39 THEN 2080
2070  GOTO 3000: REM FINISHED WITH LOWER RIGHT
2080  IF B = 22 THEN B = 23:Y = Y + 1: GOTO 2020
2090  IF B = 23 THEN B = 22:Y = Y + 1: GOTO 2020
2100 :
2110 :
3000  REM **--LOWER LEFT QUADRANT--**
3010 B = 0 + Z:Y = 20
3020  FOR X = B TO 17 STEP 2
3030  COLOR= 1 + C
3040  PLOT X,Y
3050  NEXT X
3060  IF Y < 39 THEN 3080
3070  GOTO 4000: REM FINISHED WITH LOWER LEFT
3080  IF B = 0 THEN B = 1:Y = Y + 1: GOTO 3020
3090  IF B = 1 THEN B = 0:Y = Y + 1: GOTO 3020
3100 :
3110 :
4000  REM **--UPPER RIGHT QUADRANT--**
4010 B = 22 + Z:Y = 0
4020  FOR X = B TO 39 STEP 2
4030 COLOR= 12 + C
4040  PLOT X,Y
4050  NEXT X
4060  IF Y < 19 THEN 4080
4070  GOTO 5000: REM FINISHED WITH UPPER RIGHT
4080  IF B = 22 THEN B = 23:Y = Y + 1: GOTO 4020
4090  IF B = 23 THEN B = 22:Y = Y + 1: GOTO 4020
4100 :
4110 :
5000  REM **--CENTER POLE--**
5010 B = 18 + Z:Y = 0
5020  FOR X = B TO 21 STEP 2
5030  COLOR= 15 + C
```

```
5040  PLOT X,Y
5050  NEXT X
5060  IF Y < 39 THEN 5080
5070  GOTO 6000: REM ALL FINISHED; BEGIN AGAIN
5080  IF B = 18 THEN B = 19:Y = Y + 1: GOTO 5020
5090  IF B = 19 THEN B = 18:Y = Y + 1: GOTO 5020
5100 :
5110 :
6000  REM **--RESET HORIZ. & VERT. POSITIONS--**
6010  IF Z = 1 THEN 8000: REM GOTO BRICKS AFTER 1 CYCLE
6020 B = 1:Y = 0:Z = 1: GOTO 250: REM BEGIN AGAIN
6030 B = 0:Y = 0:Z = 0:C = C + 1: GOTO 250: REM CHANGE COLOR; START OVER
7980 :
7990 :
8000  REM **--MOVING BRICKS & COLORS ROUTINE--**
8010 :
8020 :
8030  REM NQ & NE = VALUES FOR NEW X COORDINATE
8040  REM NM & NW = VALUES FOR NEW Y COORDINATE
8050  REM QV & WV = VALUES FOR SPEED OF BRICKS
8060  REM MV & EV = VALUES FOR SPEED OF BRICKS
8070  REM Q  &  E = VALUES FOR X COORDINATE
8080  REM M  &  W = VALUES FOR Y COORDINATE
8090 :
8100 :
8110  REM INITIAL VALUES FOR UPPER LEFT
8120 NM = 5:NQ = 10:QV = 2:MV = 1:Q = NQ:M = NM
8130 :
8140  REM INITIAL VALUES FOR LOWER RIGHT
8150 NW = 22:NE = 30:WV = 2:EV = 1:W = NW:E = NE
8160 :
8170 :
8180  COLOR= 0
8190  PLOT NQ,NM
8200 NM = M + MV:NQ = Q + QV
8210 :
8220  REM TEST BOUNDARIES & CHANGE DIRECTION
8230  IF NQ > 17 THEN NQ = 17:QV =  - QV
8240  IF NM > 19 THEN NM = 19:MV =  - MV
8250  IF NQ < 0 THEN NQ = 0:QV =  - QV
8260  IF NM < 0 THEN NM = 0:MV =  - MV
8270 :
8280  REM DELAY THEN RE-DRAW
8290  FOR W1 = 1 TO 10: NEXT W1
8300  COLOR= 2: PLOT Q,M
8310 :
8320 :
8330  COLOR= 0:Q = NQ:M = NM
```

```
8340  PLOT NW,NE
8350 NE = E + EV:NW = W + WV
8360  IF NW > 39 THEN NW = 39:WV =  - WV
8370  IF NE > 39 THEN NE = 39:EV =  - EV
8380  IF NW < 22 THEN NW = 22:WV =  - WV
8390  IF NE < 20 THEN NE = 20:EV =  - EV
8400  FOR W1 = 1 TO 10: NEXT W1
8410  COLOR= 13: PLOT W,E
8420  COLOR= 0:W = NW:E = NE
8430 :
8440 :
9000  REM **--RETURN TO MENU ROUTINE--**
9010 KEY =  PEEK ( - 16384): REM CHECK KBRD FOR PRESSED KEY
9020  IF KEY < 128 THEN 8190: REM < 128 = NO PRESSED KEY
9030  TEXT
9040  POKE  - 16368,0: REM CLEAR KBRD BUFFER
9050  PRINT  CHR$ (4);"RUN GAME.PATT.MENU"
```

Wall Blaster

```
100  REM ***--WALL.BLASTER--***
110 :
120 :
130 :
140  REM **--USER INFORMATION--**
150  TEXT : HOME
160  POKE  - 16368,0: REM CLEAR KBRD BUFFER
170  VTAB 5
180  PRINT "Do you want to play this game using"
190  PRINT
200  PRINT "keys on the Keyboard (K), Paddles (P),"
210  PRINT
220  PRINT "or Joystick (J)?"
230  PRINT
240  PRINT "Please type either ";: INVERSE : PRINT "K";: NORMAL : PRINT
    "  ";: INVERSE : PRINT "P";: NORMAL : PRINT " or ";: INVERSE : PRINT
    "J";: NORMAL : PRINT ": ";: GET METHOD$
250  IF METHOD$ = "K" OR METHOD$ = "k" OR METHOD$ = "P" OR METHOD$ = "p"
     OR METHOD$ = "J" OR METHOD$ = "j" THEN 320
260  PRINT : PRINT : INVERSE : PRINT "INCORRECT CHOICE!": NORMAL : PRINT

270 :
280  REM *--INCORRECT CHOICE--*
290  PRINT "You must press either ";: INVERSE : PRINT "K";: NORMAL : PRINT
    "  ";: INVERSE : PRINT "P";: NORMAL : PRINT " or ";: INVERSE : PRINT
    "J";: NORMAL : PRINT "!"
```

```
300  GOTO 170: REM ASK AGAIN
310 :
320  REM *-CORRECT CHOICE-*
330  IF METHOD$ = "P" OR METHOD$ = "p" OR METHOD$ = "J" OR METHOD$ = "j"
     THEN 520
340 :
350 :
360  REM **--KEYBOARD INSTRUCTIONS--**
370  HOME : VTAB 5
380  HTAB 15
390  INVERSE
400  PRINT "INSTRUCTIONS"
410  NORMAL
420  PRINT : PRINT : PRINT
430  PRINT "Use the UP and DOWN arrow keys to move"
440  PRINT
450  PRINT "the blaster. Press the SOLID APPLE key"
460  PRINT
470  PRINT "to fire the blaster."
480  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY TO CONTINUE:";: INPUT " ";L$
490  GOTO 690: REM  SET UP ROUTINE
500 :
510 :
520  REM **--PADDLE/JOYSTICK INSTRUCTIONS--**
530  HOME : VTAB 5
540  HTAB 15
550  INVERSE
560  PRINT "INSTRUCTIONS"
570  NORMAL
580  PRINT : PRINT :PRINT
590  PRINT "Use PADDLE 1 to move the blaster."
600  PRINT
610  PRINT "Press the button to fire the blaster!"
620  PRINT : PRINT : PRINT
630  PRINT "JOYSTICKS will work up or down only."
640  PRINT
650  PRINT "Press the button to fire the blaster!"
660  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY TO CONTINUE:";: INPUT " ";L$
670 :
680 :
690  REM  **--SET UP--**
700  HOME : VTAB 3
710  HTAB 15: INVERSE
720  PRINT "INFORMATION": NORMAL
730  PRINT : PRINT : PRINT
740  PRINT "OBJECT--to blast through the walls and"
```

```
750  PRINT
760  PRINT "explode the square. The last wall can-"
770  PRINT
780  PRINT "NOT be broken until one of four 'weak'"
790  PRINT
800  PRINT "spots is hit. Then any spot in that"
810  PRINT
820  PRINT "wall can be broken."
830  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY TO BEGIN PLAY:";: INPUT " ";L$
840  GR : HOME
850  COLOR= 9
860  FOR I = 0 TO 39: VLIN 0,39 AT I: NEXT I: REM  BACKGROUND
870 X = 0:Y = 10: REM  FIRST POSITION OF BLASTER
880 :
890 :
900  REM **--BRICK WALL--**
910  COLOR= 6
920  VLIN 2,36 AT 20
930  COLOR= 4
940  VLIN 2,36 AT 24
950  COLOR= 8
960  VLIN 2,36 AT 28
970 L1 =  INT ( RND (1) * 100)
980  IF L1 < 2 OR L1 > 32 THEN 970
990 R1 = L1: COLOR= 15
1000  VLIN L1,L1 + 4 AT 32
1010  VLIN R1,R1 + 4 AT 36
1020  HLIN 32,36 AT L1
1030  HLIN 32,36 AT L1 + 4
1040  COLOR= 0
1050  HLIN 33,35 AT L1 + 1
1060  HLIN 33,35 AT L1 + 2
1070  HLIN 33,35 AT L1 + 3
1080 :
1090 :
1100  REM **--WEAK SPOTS--**
1110 S1 =  INT ( RND (1) * 100)
1120  IF S1 < 2 OR S1 > 36 THEN 1110
1130 S2 =  INT ( RND (1) * 100)
1140  IF S2 < 2 OR S2 > 36 THEN 1130
1150 S3 =  INT ( RND (1) * 100)
1160  IF S3 < 2 OR S3 > 36 THEN 1150
1170 S4 =  INT ( RND (1) * 100)
1180  IF S4 < 2 OR S4 > 36 THEN 1170
1190 SECRET$ = "ON"
1200 :
1210 :
```

```
2000  REM **--BEGIN PLAY--**
2010 SA =  PEEK ( - 16286): REM STORE VALUE OF BUTTON 1/SOLID APPLE KEY
2020 OA =  PEEK ( -EM 16287): REM STORE VALUE OF BUTTON 0/OPEN APPLE KEY
2030 :
2040 :
2050  REM **--CHECK WHICH METHOD--**
2060  IF METHOD$ = "P" OR METHOD$ = "p" OR METHOD$ = "J" OR METHOD$ = "j"
    " THEN 2160: REM  SKIP OVER KEYBOARD LINES
2070 :
2080 :
2090  REM **--ARROW KEYS--**
2100 KEY =  PEEK ( - 16384): REM STORE VALUE OF KEY PRESSED
2110  IF (KEY - 128) = 10 THEN Y = Y + 1: GOTO 2270: REM DOWN ARROW, INC
    REMENT, SKIP PADDLE ROUTINE
2120  IF (KEY - 128) = 11 THEN Y = Y - 1: GOTO 2270: REM UP ARROW, DECRE
    MENT, SKIP PADDLE ROUTINE
2130  GOTO 2090: REM WAIT FOR KEY PRESS
2140 :
2150 :
2160  REM **--PADDLE POSITION--**
2170 PO =  PDL (0)
2180 :
2190 P1 =  PDL (1)
2200 :
2210  IF P1 > 127 AND P1 < 132 THEN 2240
2220  IF P1 < 131 THEN Y = Y - 1
2230  IF P1 > 131 THEN Y = Y + 1
2240  IF P0 > 127 AND P0 < 136 THEN 2270
2250  IF P0 > 136 THEN X = X + 1
2260  IF P0 < 136 THEN X = X - 1
2270  IF X < 0 THEN X = 0
2280  IF X > 39 THEN X = 39
2290 X = 0: REM  FOR THIS GAME, BLASTER STAYS IN FIRST COLUMN
2300  IF Y < 0 THEN Y = 0
2310  IF Y > 34 THEN Y = 34
2320  IF Y = PY AND X = PX THEN 2340
2330  COLOR= 9: VLIN PY,PY + 5 AT PX: COLOR= 0:MOVE$ = "OFF"
2340  VLIN Y,Y + 5 AT X
2350  IF SA > 127 OR OA > 127 THEN 2400
2360 PX = X:PY = Y
2370  GOTO 2000: REM CHECK FOR 'I', 'J", OR 'P'
2380 :
2390 :
2400  REM **--BLASTER FIRED (BUTTON PRESSED)--**
2410  IF MOVE$ = "ON" THEN VTAB 21: INVERSE : PRINT "YOU MUST MOVE BEFO
    RE YOU CAN FIRE AGAIN!": NORMAL : GOTO 2360: REM RESET, BEGIN AGAIN
2420 SHOTS = SHOTS + 1
2430  FOR V = X TO 38 STEP 2
```

```
2440  IF V = 20 AND  SCRN( V,Y + 2) <  > 9 THEN 5000: REM HIT WALL ROUTI
    NE
2450  IF V = 24 AND  SCRN( V,Y + 2) <  > 9 THEN 5000: REM HIT WALL ROUTI
    NE
2460  IF V = 28 AND  SCRN( V,Y + 2) <  > 9 THEN 5000: REM HIT WALL ROUTI
    NE
2470  IF V = 34 AND  SCRN( V,Y + 2) <  > 9 THEN 8000: REM HIT SQUARE ROU
    TINE
2480  HLIN V,V + 1 AT Y + 2
2490  FOR W = 1 TO 3: NEXT W
2500  COLOR= 9
2510  HLIN V,V + 1 AT Y + 2
2520  COLOR= 0
2530  NEXT V
2540 PX = X:PY = Y: GOTO 2000: REM  RESET, BEGIN AGAIN
2550 :
2560 :
5000  REM **--HIT THE WALL ROUTINE--**
5010  IF V = 28 AND SECRET$ = "OFF" THEN 5120
5020  IF V = 28 AND Y + 2 = S1 THEN SECRET$ = "OFF": GOTO 5120
5030  IF V = 28 AND Y + 2 = S2 THEN SECRET$ = "OFF": GOTO 5120
5040  IF V = 28 AND Y + 2 = S3 THEN SECRET$ = "OFF": GOTO 5120
5050  IF V = 28 AND Y + 2 = S4 THEN SECRET$ = "OFF": GOTO 5120
5060  IF V = 28 AND  SCRN( V,S1) <  > 9 THEN 6000: REM SOUND ROUTINE
5070  IF V = 28 AND  SCRN( V,S2) <  > 9 THEN 6000: REM SOUND ROUTINE
5080  IF V = 28 AND  SCRN( V,S3) <  > 9 THEN 6000: REM SOUND ROUTINE
5090  IF V = 28 AND  SCRN( V,S4) <  > 9 THEN 6000: REM SOUND ROUTINE
5100 :
5110 :
5120  COLOR= 9
5130  HLIN V,V + 1 AT Y + 2
5140 :
5150 :
6000  REM **--SOUND ROUTINE--**
6010 Z =  PEEK ( - 16336) -  PEEK ( - 16336) +  PEEK ( - 16336) -  PEEK
     ( - 16336) +  PEEK ( - 16336) -  PEEK ( - 16336) +  PEEK ( - 16336)
6020  FOR CLK = 1 TO 6:M =  PEEK ( - 16336): NEXT CLK
6030 :
6040 :
7000  REM **--MOVE SQUARE ROUTINE--**
7010 V = 39
7020  COLOR= 9
7030  VLIN L1,L1 + 4 AT 32
7040  VLIN R1,R1 + 4 AT 36
7050  HLIN 32,36 AT L1
7060  HLIN 32,36 AT L1 + 4
7070  HLIN 33,35 AT L1 + 1
7080  HLIN 33,35 AT L1 + 2
```

```
7090  HLIN 33,35 AT L1 + 3
7100 L1 =  INT ( RND (1) * 100)
7110  IF L1 < 2 OR L1 > 32 THEN 7100
7120 R1 = L1: COLOR= 15
7130  VLIN L1,L1 + 4 AT 32
7140  VLIN R1,R1 + 4 AT 36
7150  HLIN 32,36 AT L1
7160  HLIN 32,36 AT L1 + 4
7170  COLOR= 0
7180  HLIN 33,35 AT L1 + 1
7190  HLIN 33,35 AT L1 + 2
7200  HLIN 33,35 AT L1 + 3
7210  HOME :MOVE$ = "ON"
7220  GOTO 2520: REM  NEXT V
7230 :
7240 :
8000  REM **--HIT THE SQUARE ROUTINE--**
8010  FOR EX = 1 TO 4
8020  PRINT  CHR$ (7): REM  BELL
8030  NEXT EX
8040  COLOR= 9
8050  HLIN 32,36 AT L1
8060  HLIN 32,36 AT L1 + 4
8070  VLIN L1,L1 + 4 AT 32
8080  VLIN R1,R1 + 4 AT 36
8090  COLOR= 1
8100  HLIN 33,35 AT L1 + 1
8110  HLIN 33,35 AT L1 + 2
8120  HLIN 33,35 AT L1 + 3
8130 U = 34:DW = L1:DN = L1 + 4:U1 = 33:U2 = 32
8140 T = 32:T1 = L1 + 1
8150 B = 33:B1 = L1 + 2
8160 F = 34:D = L1 + 3
8170 F1 = 35:D1 = L1 + 2
8180  PRINT  CHR$ (7): REM  BELL
8190  COLOR=  INT ( RND (1) * 100)
8200  PLOT T,T1
8210 T = T + 1
8220  IF T > 39 THEN T = 39
8230  PLOT T,T1:T1 = T1 - 1
8240  IF T1 < 0 THEN T1 = 0
8250  PLOT T,T1
8260  COLOR=  INT ( RND (1) * 100)
8270  PLOT B,B1
8280 B = B - 1
8290  IF B < 0 THEN B = 0
8300  PLOT B,B1
8310 B1 = B1 + 1
```

```
8320  IF B1 > 39 THEN B1 = 39
8330  PLOT B,B1
8340  COLOR=  INT ( RND (1) * 100)
8350  PRINT  CHR$ (7): REM  BELL
8360  PLOT F,D
8370 F = F + 1:D = D + 1
8380  IF F > 39 THEN F = 39
8390  IF D > 39 THEN D = 39
8400  COLOR=  INT ( RND (1) * 100)
8410  PLOT F1,D1
8420 F1 = F1 - 1:D1 = D1 - 1
8430  IF F1 < 0 THEN F1 = 0
8440  IF D1 < 0 THEN D1 = 0
8450  COLOR=  INT ( RND (1) * 100)
8460  PLOT U,DW
8470 DW = DW - 1
8480  IF DW < 0 THEN DW = 0
8490  COLOR=  INT ( RND (1) * 100)
8500  PLOT U,DN
8510  PRINT  CHR$ (7): REM  BELL
8520 DN = DN + 1
8530  IF DN > 39 THEN DN = 39
8540  COLOR=  INT ( RND (1) * 100)
8550  PLOT U1,L1 + 2
8560 U1 = U1 - 1
8570  IF U1 < 0 THEN U1 = 0
8580  COLOR=  INT ( RND (1) * 100)
8590  PLOT U2,L1 + 2
8600 U2 = U2 + 1
8610  IF U2 > 39 THEN U2 = 39
8620 CT = CT + 1
8630  IF CT = 5 THEN 9000
8640  GOTO 8180: REM REPEAT 5 TIMES
8650 :
8660 :
9000  REM **--GIVE SCORE  START OVER/QUIT--**
9010  HOME
9020  VTAB 21
9030  POKE  - 16368,0: REM CLEAR KBRD BUFFER
9040  PRINT "YOU DID IT WITH ";: INVERSE : PRINT SHOTS;: NORMAL : PRINT
    " SHOTS!!"
9050  PRINT
9060  PRINT "PRESS THE ";: INVERSE : PRINT "PADDLE BUTTON";: NORMAL : PRINT
    " TO PLAY AGAIN."
9070  PRINT "PRESS THE ";: INVERSE : PRINT "ESC";: NORMAL : PRINT " KEY
    TO QUIT.";
9080 SA =  PEEK ( - 16286): REM STORE VALUE OF BUTTON 1/SOLID APPLE KEY
9090 OA =  PEEK ( - 16287): REM STORE VALUE OF BUTTON 0/OPEN APPLE KEY
```

```
9100 ESC =  PEEK ( - 16384) - 128: REM CHECK FOR ESC KEY PRESSED
9110  IF ESC = 27 THEN 10000: REM  END
9120  IF SA > 127 OR OA > 127 THEN  RUN : REM  PLAY AGAIN
9130  GOTO 9080: REM WAIT FOR KEY PRESS
9140 :
9150 :
10000  REM **--END--**
10010  POKE  -  16368,0: REM  CLEAR KBRD BUFFER
10020  TEXT  : HOME : VTAB 10
10030  INVERSE
10040  PRINT "THANK YOU VERY MUCH!!"
10050  NORMAL
10060  PRINT  CHR$ (4);"RUN GAME.PATT.MENU"
```

Sweeper

```
100  REM ***--SWEEPER--***
110 :
120 :
130 :
140  REM **--USER INFORMATION--**
150  TEXT : HOME : VTAB 3
160  POKE  - 16368,0: REM CLEAR KBRD BUFFER
170  INVERSE
180  HTAB 15
190  PRINT "SWEEPER"
200  NORMAL
210  VTAB 8
220  PRINT "Do you want to play this game using"
230  PRINT
240  PRINT "keys on the Keyboard (K), Paddles (P),"
250  PRINT
260  PRINT "or Joystick (J)?"
270  PRINT
280  PRINT "Please type either ";: INVERSE : PRINT "K";: NORMAL : PRINT
    "  ";: INVERSE : PRINT "P";: NORMAL : PRINT " or ";: INVERSE : PRINT
    "J";: NORMAL : PRINT ": ";: GET METHOD$
290  IF METHOD$ = "K" OR METHOD$ = "k" OR METHOD$ = "P" OR METHOD$ = "p"
     OR METHOD$ = "J" OR METHOD$ = "j" THEN 360
300  PRINT : PRINT : INVERSE : PRINT "INCORRECT CHOICE!": NORMAL : PRINT

310 :
320  REM *--INCORRECT CHOICE--*
330  PRINT "You must press either ";: INVERSE : PRINT "K";: NORMAL : PRINT
    "  ";: INVERSE : PRINT "P";: NORMAL : PRINT " or ";: INVERSE : PRINT
    "J";: NORMAL : PRINT "!"
```

```
340  GOTO 210: REM ASK AGAIN
350 :
360  REM *-CORRECT CHOICE-*
370  IF METHOD$ = "P" OR METHOD$ = "p" OR METHOD$ = "J" OR METHOD$ = "j"
     THEN 550
380 :
390 :
400  REM **--KEYBOARD INSTRUCTIONS--**
410  HOME : VTAB 3
420  HTAB 15
430  PRINT "INSTRUCTIONS"
440  NORMAL
450  VTAB 8
460  PRINT "Use the U, D, L, R, ";: INVERSE : PRINT "ARROW KEYS";: NORMAL
    : PRINT " to move"
470  PRINT
480  PRINT "the SWEEPER. Press the ";: INVERSE : PRINT "SOLID APPLE";: NORMAL
    : PRINT " key"
490  PRINT
500  PRINT "to turn the sound on or off."
510  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY TO CONTINUE:";: INPUT " ";L$
520  GOTO 1000: REM SET UP ROUTINE
530 :
540 :
550  REM **--PADDLE/JOYSTICK INSTRUCTIONS--**
560  HOME : VTAB 3
570  HTAB 15
580  PRINT "INSTRUCTIONS"
590  NORMAL
600  VTAB 7
610  PRINT "Use ";: INVERSE : PRINT "PADDLE 1";: NORMAL : PRINT " to mov
   e the SWEEPER."
620  PRINT
630  PRINT "vertically. Use ";: INVERSE : PRINT "PADDLE 0";: NORMAL : PRINT
    " to move horiz."
640  PRINT : PRINT
650  INVERSE : PRINT "JOYSTICKS";: NORMAL : PRINT " will work as they no
    rmally"
660  PRINT
670  PRINT "do."
680  PRINT : PRINT
690  PRINT "Press the ";: INVERSE : PRINT "SOLID APPLE";: NORMAL : PRINT
    " key to turn the"
700  PRINT
710  PRINT  "sound on/off."
```

```
720  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY TO CONTINUE:";: INPUT " ";L$
730 :
740 :
1000  REM **--SET UP--**
1010  HOME : VTAB 3
1020  HTAB 15: INVERSE
1030  PRINT "INFORMATION"
1040  NORMAL
1050  PRINT : PRINT
1060  PRINT "OBJECT--to sweep up all of the spots"
1070  PRINT
1080  PRINT "as fast as possible. When you have"
1090  PRINT
1100  PRINT "finished sweeping, press the 'ESC' key"
1110  PRINT
1120  PRINT "to indicate that you are done. A very"
1130  PRINT
1140  PRINT "rough approximation of your time (in"
1150  PRINT
1160  PRINT "seconds if sound was turned on) will"
1170  PRINT
1180  PRINT "then be shown."
1190  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY TO BEGIN PLAY:";: INPUT " ";L$
1200 :
1210 :
1500  REM **--CREATE SPOTS--**
1510  GR : HOME :SOUND$ = "ON"
1520 TIME = 0
1530  FOR I = 1 TO 100
1540  COLOR=  INT ( RND (1) * 100)
1550 X =  INT ( RND (1) * 100)
1560  IF X < 0 OR X > 39 THEN 1550
1570 Y =  INT ( RND (1) * 100)
1580  IF Y < 0 OR Y > 39 THEN 1570
1590  PLOT X,Y
1600  NEXT I
1610 :
1620 :
1630  REM **--INFO.DISP ON BOTTOM 4 LINES--**
1640  HOME : VTAB 21
1650  IF METHOD$ = "K" OR METHOD$ = "k" THEN KEY$ = "ARROW KEYS": GOTO 1
    680
1660  IF METHOD$ = "P" OR METHOD$ = "p" THEN KEY$ = "PADDLES": GOTO 1680
```

```
1670 KEY$ = "JOYSTICK"
1680  PRINT "Use ";: INVERSE : PRINT KEY$;: NORMAL : PRINT " to move the
     SWEEPER."
1690  PRINT "Use ";: INVERSE : PRINT "SOLID APPLE";: NORMAL : PRINT " to
     turn sound on/off."
1700  PRINT "Use ";: INVERSE : PRINT "ESC";: NORMAL : PRINT " key to qui
    t!";
1710 :
1720 :
1730 :
1740 :
2000  REM **--BEGIN PLAY--**
2010 SA =  PEEK ( - 16286): REM STORE VALUE OF BUTTON 1/SOLID APPLE KEY
2020 OA =  PEEK ( - 16287): REM STORE VALUE OF BUTTON 0/OPEN APPLE KEY
2030 :
2040 :
2050  REM **--CHECK WHICH METHOD--**
2060  IF METHOD$ = "P" OR METHOD$ = "p" OR METHOD$ = "J" OR METHOD$ = "j
    " THEN 2190: REM SKIP OVER KEYBOARD LINES
2070 :
2080 :
2090  REM **--ARROW KEYS--**
2100 KEY =  PEEK ( - 16384): REM STORE VALUE OF KEY PRESSED
2110  IF (KEY - 128) = 8 THEN X = X - 1: GOTO 2350: REM LEFT ARROW, INCR
    EMENT, SKIP PADDLE/JOYSTICK POSITION
2120  IF (KEY - 128) = 21 THEN X = X + 1: GOTO 2350: REM RIGHT ARROW, IN
    CREMENT, SKIP PADDLE/JOYSTICK POSITION
2130  IF (KEY - 128) = 10 THEN Y = Y + 1: GOTO 2350: REM DOWN ARROW, INC
    R             EMENT, SKIP PADDLE/JOYSTICK POSITION
2140  IF (KEY - 128) = 11 THEN Y = Y - 1: GOTO 2350: REM UP ARROW, DECRE
    M             ENT, SKIP PADDLE/JOYSTICK POSITION
2150  IF (KEY - 128) = 27 THEN  POKE  - 16368,0: GOTO 9000: REM SCORE/QU
    IT ROUTINE
2160  GOTO 2090: REM WAIT FOR KEY PRESS
2170 :
2180 :
2190  REM **--PADDLE/JOYSTICK POSITION--**
2200  IF (KEY - 128) = 27 THEN  POKE  - 16368,0: GOTO 9000: REM SCORE/QU
    IT ROUTINE
2210 :
2220 P0 =  PDL (0)
2230 :
2240 P1 =  PDL (1)
2250 :
2260  IF P1 > 127 AND P1 < 132 THEN 2300
2270  IF P1 < 131 THEN Y = Y - 1
```

```
2280  IF P1 > 131 THEN Y = Y + 1
2290 :
2300  IF P0 > 127 AND P0 < 136 THEN 2350
2310  IF P0 > 136 THEN X = X + 1
2320  IF P0 < 136 THEN X = X - 1
2330 :
2340 :
2350  REM **--FIGURE X & Y POSITION--**
2360  IF X < 0 THEN X = 0
2370  IF X > 39 THEN X = 39: REM DO NOT GO PAST EDGE OF SCREEN
2380  REM X=0 LEFT OUT FOR THIS GAME; SWEEPER MUST MOVE HORIZ. ALSO
2390  IF Y < 0 THEN Y = 0
2400  IF Y > 34 THEN Y = 34
2410 ESC =  PEEK ( - 16384) - 128
2420  IF ESC = 27 THEN 9000
2430 TIME = TIME + 1: VTAB 24: HTAB 33: PRINT  INT (TIME / 3);
2440  IF Y = PY AND X = PX THEN 2460
2450  COLOR= 2: VLIN PY,PY + 5 AT PX: COLOR= 0:MOVE$ = "OFF"
2460  VLIN Y,Y + 5 AT X
2470 SA =  PEEK ( - 16286): REM STORE VALUE OF BUTTON 1/SOLID APPLE KEY
2480 OA =  PEEK ( - 16287): REM STORE VALUE OF BUTTON 0/OPEN APPLE KEY
2490  IF SA > 127 THEN  GOSUB 3000: REM TURN SOUND$ ON/OFF
2500 PX = X:PY = Y
2510  IF SOUND$ = "ON" THEN  GOSUB 6000: REM SOUND ROUTINE
2520  GOTO 2000: REM CHECK FOR 'I', 'J", OR 'P'
2530 :
2540 :
2550 :
2560 :
3000  REM **--TURN SOUND ON/OFF--**
3010  IF SOUND$ = "ON" THEN SOUND$ = "OFF": RETURN
3020  IF SOUND$ = "OFF" THEN SOUND$ = "ON": RETURN
3030 :
3040 :
3050 :
3060 :
6000  REM **--SOUND ROUTINE--**
6010 Z =  PEEK ( - 16336) -  PEEK ( - 16336) +  PEEK ( - 16336) -  PEEK
    ( - 16336) +  PEEK ( - 16336) -  PEEK ( - 16336) +  PEEK ( - 16336)
6020  FOR CLK = 1 TO 6:M =  PEEK ( - 16336): NEXT CLK
6030  RETURN
6040 :
6050 :
6060 :
6070 :
9000  REM **--GIVE SCORE  START OVER/QUIT--**
```

```
 9010  HOME
 9020  VTAB 21
 9030  POKE  - 16368,0: REM CLEAR KBRD BUFFER
S9040  PRINT "YOU DID IT IN APPROX.";: INVERSE : PRINT  INT (TIME / 3);:
    NORMAL : PRINT " SECOND
 9050  PRINT
 9060  PRINT "PRESS THE ";
 9070  IF METHOD$ = "K" OR METHOD$ = "k" THEN KEY$ = "SOLID APPLE": GOTO
    9090
 9080 KEY$ = "BUTTON"
 9090 : INVERSE : PRINT KEY$;: NORMAL : PRINT " TO PLAY AGAIN."
 9100  PRINT "PRESS THE ";: INVERSE : PRINT "ESC";: NORMAL : PRINT " KEY
    TO QUIT.";
 9110 SA =  PEEK ( - 16286): REM STORE VALUE OF BUTTON 1/SOLID APPLE KEY
 9120 OA =  PEEK ( - 16287): REM STORE VALUE OF BUTTON 0/OPEN APPLE KEY
 9130 ESC =  PEEK ( - 16384) - 128: REM CHECK FOR ESC KEY PRESSED
 9140  IF ESC = 27 THEN 10000: REM END
 9150  IF SA > 127 OR OA > 127 THEN 1500: REM PLAY AGAIN
 9160  GOTO 9110: REM WAIT FOR KEY PRESS
 9170 :
 9180 :
 10000  REM **--END--**
 10010  POKE  - 16368,0: REM CLEAR KBRD BUFFER
 10020  TEXT : HOME : VTAB 10
 10030  INVERSE
 10040  PRINT "THANK YOU VERY MUCH!!"
 10050  NORMAL
 10060  PRINT  CHR$ (4);"RUN GAME.PATT.MENU"
```

Maze

```
 100  REM ***--MAZE--***
 110 :
 120 :
 130 :
 140  REM **--USER INFORMATION--**
 150  TEXT : HOME
 160  POKE  - 16368,0: REM CLEAR KBRD BUFFER
 170  VTAB 5
 180  PRINT "Do you want to play this game using"
 190  PRINT
 200  PRINT "the Keyboard (K), or Joystick (J)?"
 210  PRINT
 220  PRINT "Please type either ";: INVERSE : PRINT "K";: NORMAL : PRINT
    " or ";: INVERSE : PRINT "J";: NORMAL : PRINT ": ";: GET METHOD$
```

```
230  IF METHOD$ = "K" OR METHOD$ = "k" OR METHOD$ = "J" OR METHOD$ = "j"
     THEN 300
240  PRINT : PRINT : INVERSE : PRINT "INCORRECT CHOICE!": NORMAL : PRINT

250 :
260  REM *--INCORRECT CHOICE--*
270  PRINT "You must press either ";: INVERSE : PRINT "K";: NORMAL : PRINT
    " or ";: INVERSE : PRINT "J";: NORMAL : PRINT "!"
280  GOTO 170: REM ASK AGAIN
290 :
300  REM *-CORRECT CHOICE-*
310  IF METHOD$ = "J" OR METHOD$ = "j" THEN 610
320 :
330 :
340  REM **--KEYBOARD INSTRUCTIONS--**
350  HOME
360  HTAB 15
370  PRINT "INSTRUCTIONS"
380  NORMAL
390  PRINT : PRINT
400  PRINT "Use the U, D, L, R, ";: INVERSE : PRINT "ARROW KEYS";: NORMAL
    : PRINT " to move"
410  PRINT
420  PRINT "the MAZE RUNNER. Press the ";: INVERSE : PRINT "OPEN APPLE":
     NORMAL
430  PRINT
440  PRINT "to stop the RUNNER until you are ready"
450  PRINT
460  PRINT "to continue or hold it down and use one"
470  PRINT
480  PRINT "of the ARROW KEYS to go one step at a"
490  PRINT
500  PRINT "time. Press the ";: INVERSE : PRINT "SOLID APPLE";: NORMAL :
     PRINT "key to re-"
510  PRINT
520  PRINT "position and begin again."
530  PRINT
540  PRINT "Whenever you want to quit, press the"
550  PRINT
560  INVERSE : PRINT "ESC";: NORMAL : PRINT " key."
570  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY TO CONTINUE:";: INPUT " ";L$
580  GOTO 1000: REM SET UP ROUTINE
590 :
600 :
610  REM **--JOYSTICK INSTRUCTIONS--**
620  HOME : VTAB 3
```

```
630  HTAB 15
640  INVERSE
650  PRINT "INSTRUCTIONS"
660  NORMAL
670  PRINT : PRINT
680  PRINT "JOYSTICKS will operate as they normally"
690  PRINT
700  PRINT "do."
710  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY TO CONTINUE:";: INPUT " ";L$
720 :
730 :
740 :
750 :
1000  REM **--SET UP--**
1010  HOME : VTAB 3
1020  HTAB 15: INVERSE
1030  PRINT "INFORMATION": NORMAL
1040  PRINT : PRINT : PRINT
1050  PRINT "OBJECT--to move from the starting point"
1060  PRINT
1070  PRINT "through the maze as quickly as possible"
1080  PRINT
1090  PRINT "without crashing into any of the walls."
1100  PRINT : PRINT
1110  VTAB 23: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY TO BEGIN PLAY:";: INPUT " ";L$
1120 :
1130 :
1140  REM **--DETERMINE COLOR--**
1150 C =  INT ( RND (1) * 100)
1160 C7 =  INT (C / 16)
1170 C8 = (C7 * 16) - 1
1180 C9 = C - C8
1190 C = C9
1200  IF C = 0 OR C + 1 = 0 THEN 1140
1210 :
1220 :
1230  REM **--INFO.DISP ON BOTTOM 4 LINES--**
1240  HOME : VTAB 21
1250  IF METHOD$ = "K" OR METHOD$ = "k" THEN KEY$ = "ARROW KEYS": GOTO 1
    270
1260 KEY$ = "JOYSTICK"
1270  PRINT "Use ";: INVERSE : PRINT KEY$;: NORMAL : PRINT " to move the
     RUNNER."
1280  PRINT "Use ";: INVERSE : PRINT "OPEN APPLE";: NORMAL : PRINT " key
     to pause."
```

```
1290  PRINT "Use ";: INVERSE : PRINT "SOLID APPLE";: NORMAL : PRINT " ke
    y to begin again."
1300  PRINT "Use ";: INVERSE : PRINT "ESC";: NORMAL : PRINT " key to qui
    t!";
1310 :
1320 :
1330  REM **--DRAW MAZE--**
1340  GR : COLOR= C
1350 CRASH$ = "OFF"
1360 :
1370  REM OUTSIDE LINES
1380  HLIN 12,33 AT 7
1390  HLIN 12,33 AT 28
1400  VLIN 7,22 AT 12
1410  VLIN 25,27 AT 12
1420  VLIN 7,13 AT 33
1430  VLIN 16,28 AT 33
1440 :
1450  REM INTERNAL VERT.LINES
1460  VLIN 10,18 AT 15
1470  VLIN 21,25 AT 15
1480  VLIN 10,22 AT 18
1490  VLIN 25,27 AT 18
1500  VLIN 23,25 AT 21
1510  VLIN 8,9 AT 24
1520  VLIN 14,18 AT 24
1530  VLIN 25,27 AT 24
1540  VLIN 11,12 AT 27
1550  VLIN 20,25 AT 27
1560  VLIN 25,27 AT 30
1570 :
1580  REM INTERNAL HORIZ.LINES
1590  HLIN 13,14 AT 18
1600  HLIN 16,17 AT 22
1610  HLIN 18,24 AT 10
1620  HLIN 21,27 AT 13
1630  HLIN 21,27 AT 19
1640  HLIN 19,21 AT 16
1650  HLIN 16,23 AT 22
1660  HLIN 28,32 AT 22
1670  HLIN 27,32 AT 16
1680  HLIN 27,30 AT 10
1690  HLIN 30,32 AT 13
1700  PLOT 30,19
1710 :
1720 :
1730  REM **--FIRST POSITION--**
```

```
1740  COLOR= C + 1
1750 X = 0:Y = 0: PLOT X,Y
1760 PX = X:PY = Y
1770 :
1780 :
1790 :
1800 :
2000  REM **--BEGIN PLAY--**
2010 SA =  PEEK ( - 16286): REM STORE VALUE OF BUTTON 1/SOLID APPLE KEY
2020 OA =  PEEK ( - 16287): REM STORE VALUE OF BUTTON 0/OPEN APPLE KEY
2030 :
2040 :
2050  REM **--CHECK WHICH METHOD--**
2060  IF METHOD$ = "J" OR METHOD$ = "j" THEN 2200: REM SKIP OVER KEYBOAR
    D LINES
2070 :
2080 :
2090  REM **--ARROW KEYS--**
2100 COUNT = COUNT + 1: VTAB 24: HTAB 33: PRINT COUNT;
2110 KEY =  PEEK ( - 16384): REM STORE VALUE OF KEY PRESSED
2120  IF (KEY - 128) = 8 THEN X = X - 1: GOTO 2370: REM LEFT ARROW, INCR
    EMENT, SKIP JOYSTICK ROUTINE
2130  IF (KEY - 128) = 21 THEN X = X + 1: GOTO 2370: REM RIGHT ARROW, IN
    CREMENT, SKIP JOYSTICK ROUTINE
2140  IF (KEY - 128) = 10 THEN Y = Y + 1: GOTO 2370: REM DOWN ARROW, INC
    REMENT, SKIP JOYSTICK ROUTINE
2150  IF (KEY - 128) = 11 THEN Y = Y - 1: GOTO 2370: REM UP ARROW, DECRE
    MENT, SKIP JOYSTICK ROUTINE
2160  IF (KEY - 128) = 27 THEN 2165
2162  GOTO 2170
2165  TEXT : HOME : POKE  - 16368,0: VTAB 10: INVERSE
2167  PRINT "I HOPE YOU HAD A GOOD TIME. SO LONG!": NORMAL : GOTO 10060:
     REM GO TO MENU
2170  GOTO 2090: REM WAIT FOR KEY PRESS
2180 :
2190 :
2200  REM **--JOYSTICK POSITION--**
2210 KEY =  PEEK ( - 16384): REM STORE VALUE OF KEY PRESSED
2220  IF (KEY - 128) = 27 THEN 2225
2222  GOTO 2230
2225  TEXT : HOME : POKE  - 16368,0: VTAB 10: INVERSE
2227  PRINT "I HOPE YOU HAD A GOOD TIME. SO LONG!": NORMAL : GOTO 10060:
     REM  GO TO MENU
2230 COUNT = COUNT + 1: VTAB 24: HTAB 33: PRINT COUNT;
2240 :
2250 P0 =  PDL (0)
2260 :
2270 P1 =  PDL (1)
```

```
2280 :
2290  IF P1 > 127 AND P1 < 132 THEN 2320
2300  IF P1 < 131 THEN Y = Y - 1
2310  IF P1 > 131 THEN Y = Y + 1
2320  IF P0 > 127 AND P0 < 136 THEN 2370
2330  IF P0 > 136 THEN X = X + 1
2340  IF P0 < 136 THEN X = X - 1
2350 :
2360 :
2370  REM **--FIGURE X & Y POSITION--**
2380  IF X < 0 THEN X = 0
2390  IF X > 39 THEN X = 39
2400  REM X=0 LEFT OUT FOR THIS GAME; RUNNER MUST MOVE HORIZ. ALSO
2410  IF Y < 0 THEN Y = 0
2420  IF Y > 39 THEN Y = 39
2430  IF Y = PY AND X = PX THEN 2490
2440 :
2450  REM BLACKOUT RUNNER'S CURRENT POSITION
2460  COLOR= 0: PLOT PX,PY: COLOR= C + 1
2470  FOR I = 1 TO 5: NEXT I
2480  IF  SCRN( X,Y) = C THEN CRASH$ = "ON": REM CRASHED
2490  PLOT X,Y
2500  IF SCRN( 34,14) = C + 1 OR  SCRN( 34,15) = C + 1 THEN 9000: REM A
    LL THRU, WINNER!
2510  IF CRASH$ = "ON" THEN 5000: REM GO TO CRASH ROUTINE
2520 PX = X:PY = Y
2530  IF SA > 127 THEN 3000: REM GO TO BEGIN AGAIN ROUTINE
2540  IF OA > 127 THEN 4000: REM GO TO PAUSE ROUTINE
2550 PX = X:PY = Y
2560  GOTO 2000: REM CHECK FOR 'I', OR 'J"
2570 :
2580 :
2590 :
2600 :
3000  REM **--BEGIN AGAIN ROUTINE--**
3010 C =  INT ( RND (1) * 100)
3020 C7 =  INT (C / 16)
3030 C8 = (C7 * 16) - 1
3040 C9 = C - C8
3050 C = C9
3060 X = 0:Y = 0
3070 COUNT = 0
3080  COLOR= C
3090  GOTO 1230: REM DISPLAY INFO
3100 :
3110 :
3120 :
3130 :
```

```
4000  REM **--PAUSE ROUTINE--**
4010  POKE  - 16368,0: REM CLEAR KBRD BUFFER
4020 KEY =  PEEK ( - 16384): REM STORE VALUE OF PRESSED KEY
4030 SA =  PEEK ( - 16286): REM STORE VALUE OF BUTTON 1/SOLID APPLE KEY
4040 OA =  PEEK ( - 16287): REM STORE VALUE OF BUTTON 0/OPEN APPLE KEY
4050  IF (KEY - 128) = 27 THEN 4055
4052  GOTO 4060
4055  TEXT : HOME : POKE  - 16368,0: VTAB 10: INVERSE
4057  PRINT "I HOPE YOU HAD A GOOD TIME. SO LONG!": NORMAL : GOTO 10060:
     REM   GO TO MENU
4060 COUNT = COUNT + 1: VTAB 24: HTAB 33: PRINT COUNT;
4070  IF KEY > 127 OR OA > 127 OR SA > 127 THEN PX = X:PY = Y: GOTO 2000
    : REM RESET & BEGIN PLAY AGAIN
4080  GOTO 4020: REM WAIT FOR KEY PRESS
4090 :
4100 :
4110 :
4120 :
5000  REM **--CRASH ROUTINE--**
5010  COLOR= 1
5020  FOR I = 1 TO 5
5030  PLOT X - I,Y - I
5040  PRINT  CHR$ (7): REM BELL
5050  NEXT I
5060  HOME
5070  VTAB 21
5080  PRINT "SORRY. YOU CRASHED THIS TIME!"
5090 :
5100 :
5980 :
5990 :
9000  REM **--GIVE SCORE  START OVER/QUIT--**
9010  IF CRASH$ = "ON" THEN 9030
9020  HOME
9030  VTAB 22
9040  POKE  - 16368,0: REM CLEAR KBRD BUFFER
9050  IF CRASH$ = "ON" THE 9090
9060  FOR I = 1 TO 5: PRINT  CHR$ (7); CHR$ (7): NEXT I
9070  PRINT "YOU DID IT IN A COUNT OF ";: INVERSE : PRINT COUNT;: NORMAL

9080  PRINT
9090  PRINT "PRESS THE ";: INVERSE
9100  IF METHOD$ = "K" OR METHOD$ = "k" THEN KEY$ = "SOLID APPLE": GOTO
    9120
9110 KEY$ = "BUTTON"
9120  PRINT KEY$;: NORMAL : PRINT " TO PLAY AGAIN."
9130  PRINT "PRESS THE ";: INVERSE : PRINT "ESC";: NORMAL : PRINT " KEY
    TO QUIT.";
```

```
9140 SA =  PEEK ( - 16286): REM STORE VALUE OF BUTTON 1/SOLID APPLE KEY
9150 OA =  PEEK ( - 16287): REM STORE VALUE OF BUTTON 0/OPEN APPLE KEY
9160 ESC =  PEEK ( - 16384) - 128: REM CHECK FOR ESC KEY PRESSED
9170  IF ESC = 27 THEN 10000: REM END
9180  IF SA > 127 OR OA > 127 THEN 3000: REM PLAY AGAIN
9190  GOTO 9140: REM WAIT FOR KEY PRESS
9200 :
9210 :
10000  REM **--END--**
10010  POKE  - 16368,0: REM CLEAR KBRD BUFFER
10020  TEXT : HOME : VTAB 10
10030  INVERSE
10040  PRINT "THANK YOU VERY MUCH!!"
10050  NORMAL
10060  PRINT  CHR$ (4);"RUN GAME.PATT.MENU"
```

Free Drawing

```
100  REM  ***--FREE.DRAWING--***
110 :
120 :
130  HOME : GR
140  COLOR=  INT ( RND (1) * 100)
150 X = 10:Y = 20
160  PLOT X,Y
170 OA =  PEEK ( - 16287)
180  FOR I = 1 TO 5: NEXT I
190 P0 =  PDL (0)
200  FOR I = 1 TO 5: NEXT I
210 P1 =  PDL (1)
220  FOR I = 1 TO 5: NEXT I
230  IF P1 > 120 AND P1 < 132 THEN 260
240  IF P1 < 131 THEN Y = Y - 1
250  IF P1 > 131 THEN Y = Y + 1
260  IF P0 > 120 AND P0 < 136 THEN 290
270  IF P0 > 136 THEN X = X + 1
280  IF P0 < 136 THEN X = X - 1
290  IF X < 0 THEN X = 0
300  IF X > 39 THEN X = 39
310  IF Y < 0 THEN Y = 0
320  IF Y > 39 THEN Y = 39
330  PLOT X,Y
340  IF OA > 127 THEN 10000: GR : COLOR=  INT ( RND (1) * 100):X = 20:Y =
   20
350  GOTO 170
360 :
```

```
370 :
10000  REM **--END--**
10010  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
10020  HOME : VTAB 21
10030  VTAB 21
10040  INPUT "DO YOU WANT TO SAVE THIS DRAWING (Y/N)";Q$
10050  IF Q$ = "Y" OR Q$ = "y" OR Q$ = "N" OR Q$ = "n" THEN 10070
10060  GOTO 10030: REM  ASK AGAIN
10070  IF Q$ = "N" OR Q$ = "n" THEN 10200: REM END
10080  PRINT "WHAT DO YOU WANT TO CALL THIS DRAWING?"
10085  PRINT
10090  INPUT "TYPE NAME FOR DRAWING: ";DRW$
10095  HOME : VTAB 23
10100  PRINT  CHR$ (4);"BSAVE" + DRW$ + ",A$400, L$3FF"
10200  PRINT  CHR$ (4);"RUN GAME.PATT.MENU"
```

View Drawing

```
100  REM  ***--VIEW.DRAWING--***
110 :
120 :
130 D$ =  CHR$ (4): REM  CONTROL D
140  TEXT : HOME
145  PRINT D$;"CATALOG"
147  PRINT
150  PRINT "Which drawing would you like to see? "
160  PRINT
170  INPUT "Type the drawing's name: ";DRW$
180  HOME : GR
190  PRINT D$;"BLOAD";DRW$
200  VTAB 21
210  INPUT "Do you want to see another one (Y/N) ";Q$
220  IF Q$ = "Y" OR Q$ = "y" OR Q$ = "N" OR Q$ = "n" THEN 240
230  GOTO 200: REM  ASK AGAIN
240  IF Q$ = "Y" OR Q$ = "y" THEN  RUN
250  PRINT D$;"RUN GAME.PATT.MENU"
```

CHAPTER 7

High-Resolution Graphics

As you begin to study high-resolution graphics, it is important to remember that this book is an introductory book, not a comprehensive study of any one aspect of graphics. I intend to present enough information here in an understandable manner that you will be able to use Applesoft BASIC to create useful and enjoyable high-resolution graphics. The high-resolution chapters are not going to teach you enough to create commercial-quality animated arcade games or Computer Aided Design programs. These chapters are designed to be an introduction to high-resolution graphics using BASIC as the main vehicle.

HIGH-RESOLUTION SCREENS

Both high-resolution screens can contain the same amount of information (i.e., they can both have a 280 by 192 size screen). Both screens use the same type of coordinates the low-resolution screen uses. The upper left point is 0,0 and the lower right point is 279,191 (Fig. 7.1). Therefore, if you have learned to use low-resolution, you should not have too much difficulty learning the coordinate system of the high-resolution screens.

Confusion can occur, however, regarding the location of the coordinates in the BASIC high-resolution command statements. In low-resolution commands, you use different instructions to inform the computer that you want to draw a horizontal line (HLIN), draw a vertical line (VLIN), or plot a single point (PLOT). But high-resolution has a single command that can accomplish all three tasks.

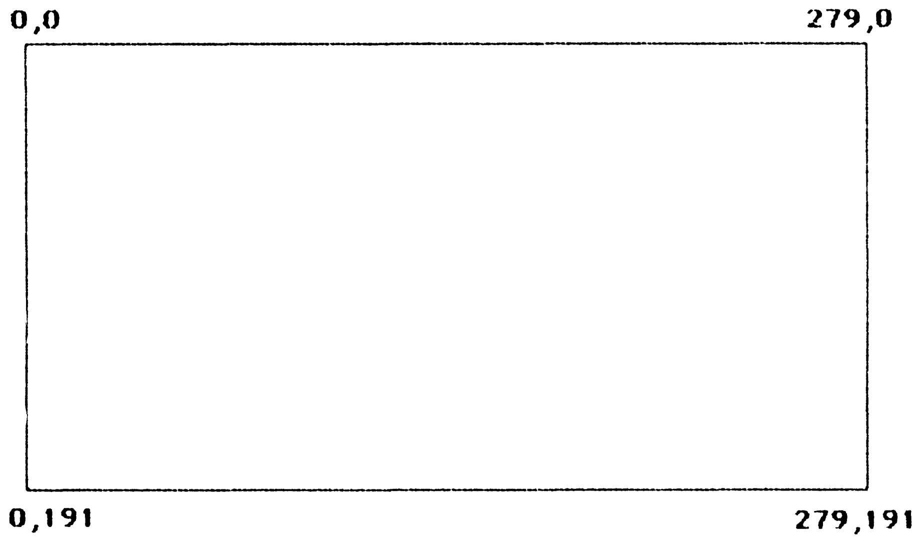


FIGURE 7.1. High-resolution screen dimensions.

HIGH-RESOLUTION SINGLE POINT

The HPLOT command can plot from a specified point to another specified point. You must provide the coordinates for each of the specified points. Each point is defined by its location on the screen using its column and row value. Remember that the columns in high-resolution are numbered from 0 to 279 and the rows are numbered from 0 to 191. Therefore, any point can be defined by providing its column and row value (i.e., 20,5 specifies a point on the 20th column and the 5th row; Fig. 7.2). To plot this point on the high-resolution screen, you must issue the command

```
HPLOT 20,5 {RETURN}
```

Remember to provide the column value first and then the row value.

Before issuing the HPLOT command, you must place the computer in one of its two high-resolution modes by using either the HGR or HGR2 command, and then you must specify one of the eight high-resolution colors. For example:

```
HGR {RETURN}

HCOLOR = 6 {RETURN}

HPLOT 20,5 {RETURN}
```

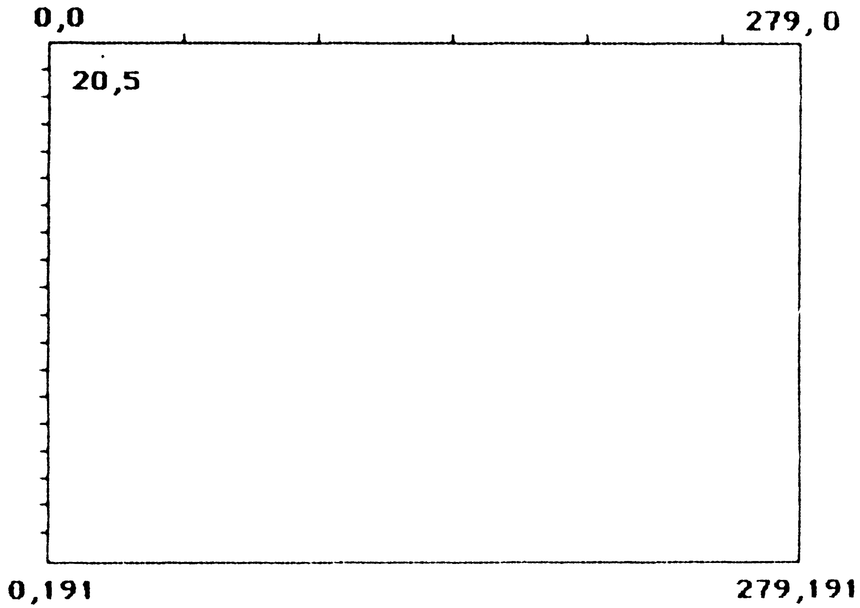


FIGURE 7.2. Single point (5th row, 20th column).

HIGH-RESOLUTION HORIZONTAL LINE

If you want to draw a horizontal line from the 15th column to the 25th column on the 5th row, you must specify the starting point and the ending point of the line. The starting point is at column 15, row 5 (15,5) and the ending point is at column 25, row 5(25,5) (Fig. 7.3). To plot this horizontal line on the high-resolution screen, you issue the command

```
HPLOT 15,5 TO 25,5 {RETURN}
```

HIGH-RESOLUTION VERTICAL LINE

To plot a vertical line, follow the same format. Determine the coordinates of the starting point and the ending point, and include those values in an HPLOT statement. If you want to draw a vertical line from the 5th row to the 35th row on the 20th column, the starting point is at column 20, row 5 (20,5) and the ending point is at column 20, row 35(20,35) (Fig. 7.4). To plot this vertical line on the high-resolution screen, you issue the command

```
HPLOT 20,5 TO 20,35 {RETURN}
```

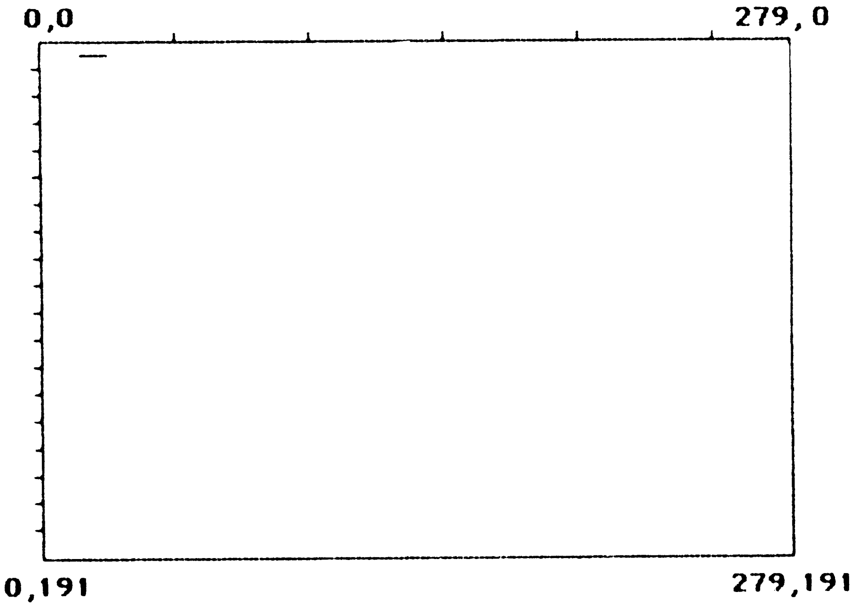


FIGURE 7.3. Horizontal line from row 5, column 15, to row 5, column 25.

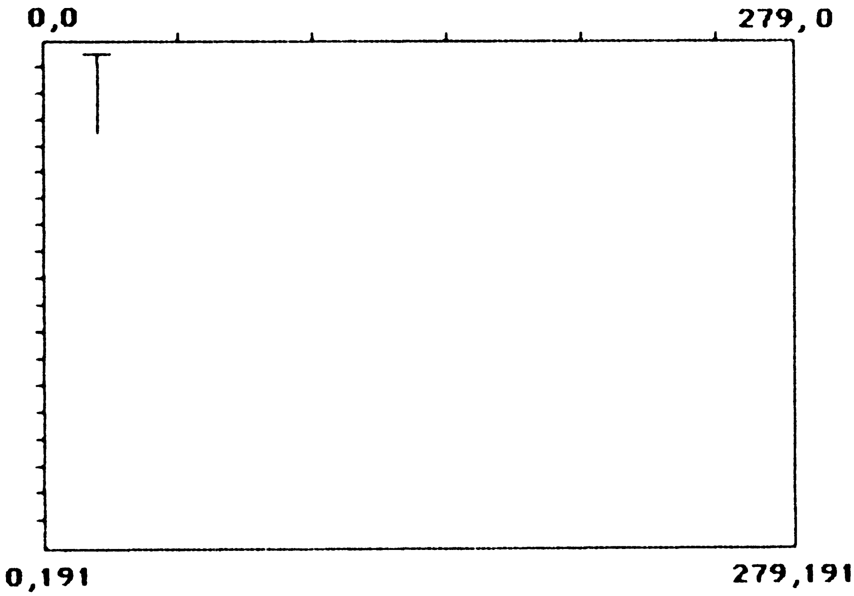


FIGURE 7.4. Vertical line added from row 5, column 20, to row 35, column 20.

With this information, you should be able to convert the LOW.RES.I program from Chapter 6 to a high-resolution version of the same program. I encourage you to attempt the

conversion on your own, but if you need help, I have included a listing of the conversion at the end of this chapter (the listing is called HI.RES.I).

HIGH-RESOLUTION DIAGONAL LINE

Diagonal lines are drawn in the same manner as horizontal and vertical lines: determine the coordinates of the starting point and the ending point, and include these values in an HPLOT statement. If you want to draw a diagonal line from a point at the 50th column on row 25 to the 100th column on row 75, you must specify the starting point and the ending point of the line. The starting point is at column 50, row 25 (50,25) and the ending point is at column 100, row 75 (100,75) (Fig. 7.5). To plot this diagonal line on the high-resolution screen, you issue the command

```
HPLOT 50,25 TO 100,75 {RETURN}
```

As you can see, the instructions for horizontal lines, vertical lines and diagonal lines are very similar:

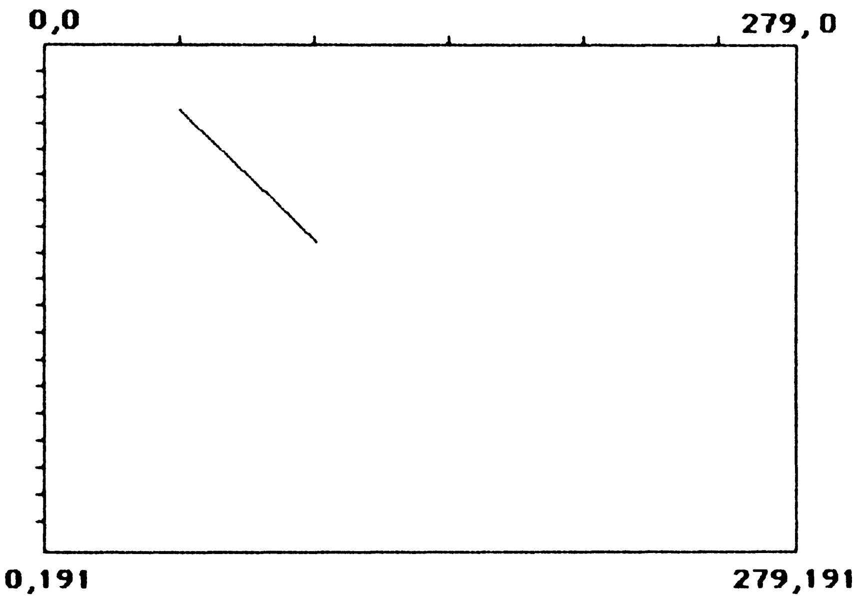


FIGURE 7.5. Diagonal line from row 25, column 50, to row 75, column 100.

```
HPLOT 15,5 TO 25,5

HPLOT 20,5 TO 20,35

HPLOT 50,25 TO 100,75
```

To help distinguish between these instructions, you can use REM to indicate the function of each instruction:

```
HPLOT 15,5 TO 25,5:REM HORIZONTAL LINE

HPLOT 20,5 TO 20,35: REM VERTICAL LINE

HPLOT 50,25 TO 100,75: REM DIAGONAL LINE
```

HIGH-RESOLUTION RECTANGLE

A single HPLOT instruction can be used to draw a series of lines as long as those lines are connected to each other. For example, a rectangle can be drawn by specifying the starting point of the first line, the ending point of the first line and the ending point of each subsequent line (assuming therefore, that the ending point of one line is the starting point of the next line). A rectangle whose coordinates are

upper left—100,50
upper right—200,50
lower right—200,125
lower left—100,125

can be drawn (Fig. 7.6) with the following instruction:

```
HPLOT 100,50 TO 200,50 TO 200,125 TO 100,125 TO 100,50
```

The same rectangle can, of course, be drawn with four separate HPLOT instructions:

```
HPLOT 100,50 TO 200,50:REM TOP HORIZ.

HPLOT 200,50 TO 200,125:REM RIGHT VERT.

HPLOT 200,125 TO 100,125:REM BOTT.HORIZ.

HPLOT 100,125 TO 100,50:REM LEFT VERT.
```

The instructions for this rectangle draw the lines in the following order: the top line, the right side, the bottom line and finally the left side. Obviously, a rectangle can be drawn in a different order. If the order is different, the instructions will be somewhat different even

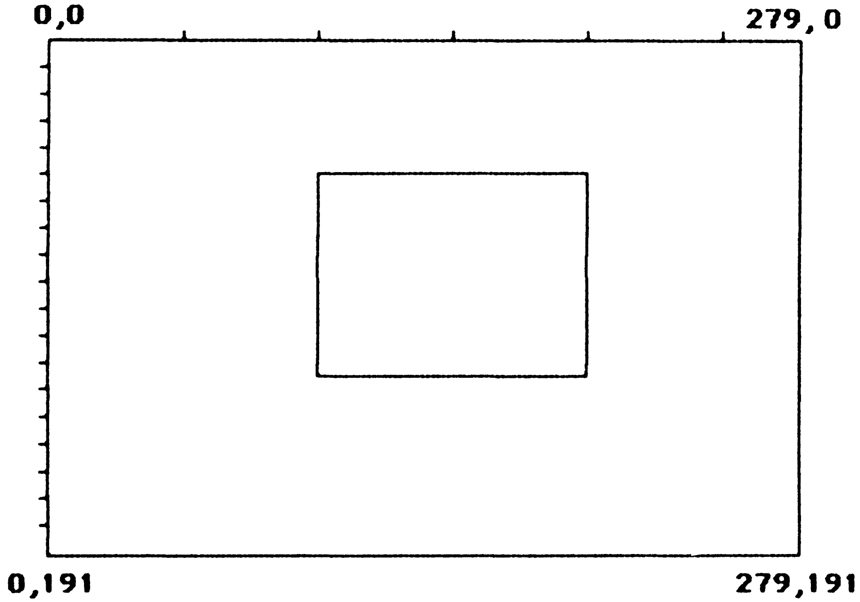


FIGURE 7.6. High-resolution rectangle.

though the resulting rectangle will be the same. For example, the following instruction draws the same rectangle as before:

```
HPLOT 100,125 TO 200,125 TO 200,50 TO 100,50 TO 100,125
```

The two instructions use the same coordinates but in a different order. The lines still all connect, but this time I started with the bottom line, went to the right side, then the top line and finally the left side.

Although the computer clearly understands all three of these sets of instructions to draw this rectangle, someone reading the program listing will have a difficult time figuring out the intent of the instructions unless the programmer has included remarks. For long lines of instructions, using REM at the end of the line may not be practical. Therefore, a separate instruction line containing only the remark may be the best idea.

```
205 REM **--DRAW RECTANGLE/BEG.TOP RT--**

210 HPLOT 100,50 TO 200,50 TO 200,125 TO 100,125 TO 100,50
```

or

```
205 REM **--DRAW RECTANGLE/BEG.LOW LEFT--**
```

```
210 HPLOT 100,125 TO 200,125 TO 200,50 TO 100,50 TO 100,125
```

However you choose to do it, it is important to include some indication of the intent of the HPLOT instruction in a REM statement, so a reader can visualize what your HPLOT instructions are meant to do.

HIGH-RESOLUTION CIRCLES

Drawing circles in high-resolution usually involves more than simple HPLOT instructions. You could graph each point on a circle and then specify the coordinates for those points in a series of HPLOT statements, but the time and effort involved would be considerable. It is simpler to use a routine that contains some math functions that figure out what the coordinates of each point on the circle are. You do not need to understand the math functions in order to use the circle routine. All you do need to know is which variables to change in order to make the circle the way you want it. The routine presented below is fairly common.

```
100 REM ***--HI.RES.CIRCLE--***
110:
120:
130 REM **--INITIAL VALUES--**
140 PI = 3.1415926::REM VALUE FOR PI
150 COLUMN1 = 140:::REM FIRST COLUMN POSITION
160 ROW1 = 80:::::::REM FIRST ROW POSITION
170 RADIUS = 70:::::REM SIZE OF CIRCLE
180 NUMBER = 256::::REM NUMBER OF POINTS ON CIRCLE
190 CC = 2*PI:::::::REM EXTENT OF LOOP
200 INC = CC/NUMBER:REM INCREMENT IN LOOP
210:
220:
500 REM **--SET UP GRAPHICS--**
510 HGR
520 HCOLOR = 3:REM USE WHITE TO DISPLAY ALL PTS
530:
```

```
540:
700 REM **--DRAW CIRCLE--**
710 FOR I = 0 TO CC STEP INC
720 CLMN = RADIUS * SIN(I)
730 ::RW = RADIUS * COS(I)
735 RW = RW *.83:REM ADJUSTMENT FOR ELLIPSE EFFECT
740 HPLOT COLUMN1 + CLMN,ROW1 + RW
750 NEXT I
760:
770:
800 REM **--END--**
810 END
```

This routine will produce a white circle on the first high-resolution screen by plotting points counterclockwise beginning at point 140,80 (column 140, row 80). 256 points are plotted for the circumference of this circle. The circle has a radius of 70 points (Fig. 7.7). By changing the first column and/or first row position, you can change the location of the circle (i.e., changing the values of COLUMN1 and/or ROW 1). By changing the value of the radius, you can change the size of the circle. By changing the number of points on the circle, you can change the resolution of the circle. The value in line 735 can be changed to reflect a more perfect circle depending upon the display device you are using. Each display device I have

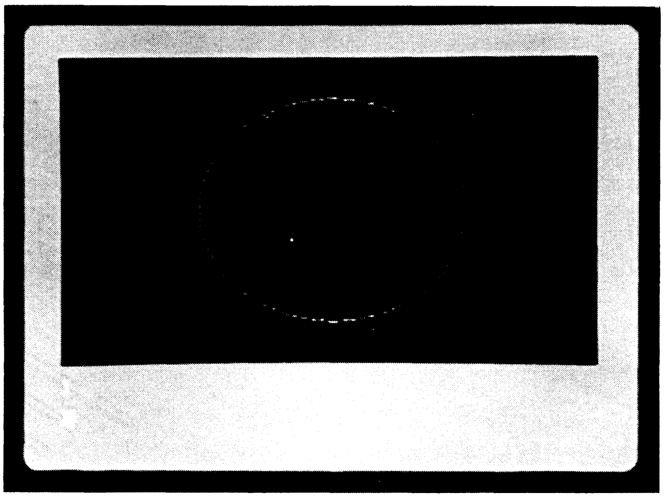


FIGURE 7.7. High-resolution circle.

used required a slightly different number in line 735. The monitor I am using with the Apple *II*e required a value in line 735 of .83, while the Apple IIGS monitor required a value of .95. The value may range from as low as .66 to as high as 1.00.

The extra colons are simply used for formatting purposes. I think that lining up all the REM's makes it easier to read and perhaps understand. The actual HI.RES.CIRCLE routine is a short FOR-NEXT loop that makes use of the trigonometric functions sine and cosine (SIN and COS). Again, it is not necessary for you to understand these functions or even how they accomplish the task of determining the coordinates of a circle. All that is necessary is that you know that these functions work and that they can be used to create all kinds of circles or curved lines. You can experiment with these two math functions to produce some very nice high-resolution patterns (Fig. 7.8). I have included some examples at the end of this chapter.

BACKGROUND COLOR

It is not necessary to do all graphic work on a black background. Apple provides a routine that allows you to change the background to any of the available high-resolution colors. The program listed below demonstrates how the background color can be changed.

```
100 REM ***--BACKGROUND.DEMO--**
110:
120:
```

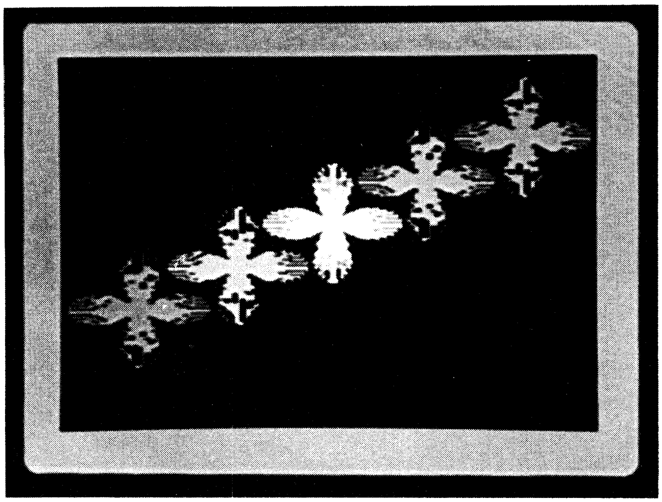


FIGURE 7.8. Polar roses.

```
130 REM **--INITIALIZE GRAPHICS--**
140 HGR
150:
160:
200 REM **--CHANGE BACKGROUND--**
210 FOR COLR = 0 to 7
220 HCOLOR = COLR
230 HPLOT 0,0
240 CALL -3082:REM CLEARS SCRN TO LAST COLOR PLOTED
250 INPUT "";L$:REM WAIT UNTIL READY TO CONTINUE
260 NEXT COLR
270:
280:
300 REM **--END--**
310 TEXT
320 END
```

The CALL −3082 routine clears the high-resolution screen to the last color plotted. If you simply want to quickly clear the high-resolution screen to black, Apple provides another routine, CALL −3086. In addition to these two routines specific to high-resolution graphics, there are a number of others that are available to BASIC programmers. I have included a list of some at the end of this chapter, but there are reference books available that provide much more information on this subject.

ANIMATION

Animation is the process of moving something around on the screen—rather it is the process of making the object *appear* to move. So far in this chapter we have seen how we can plot a point, draw a horizontal line, draw a vertical line, draw a diagonal line, create a rectangle, create a circle, and change the background color of the high-resolution screen. Now we will see how programmers actually animate objects in high-resolution.

Most animation is not done using BASIC, because BASIC is an interpreted language and therefore runs slower than assembly language. Whenever a BASIC program is run, the computer must translate the instructions into machine language—hexadecimal values—as the program is operating. Assembly language is compiled into hexadecimal values before it

is run and therefore does not need to be translated as the program is operating. The translation process significantly slows BASIC down and means that animation done using BASIC will not appear as smooth or as fast as animation created using assembly language. But the process of producing animation remains much the same in any language.

To create an object that moves on the screen, you must first create the object in a stationary state. In other words, to begin with, you must position the object somewhere on the screen. To make the object appear to change position, you erase the object from its current position and re-create it in a new location. If the object must change positions ten different times, each time the current position must be erased and the object recreated in a new position. The more complex the object is, the more difficult it is to create the first time, and the more time it will take to re-create it each time it changes position.

To demonstrate the process of animation, we will use the simple object we have been using so far, a capital letter "I". As stated, the first step is to create the object. The following instructions accomplish this:

```
100 REM ***--ANIMATED.I--**
110:
120:
130 REM **--INITIALIZE HI-RES GRAPHICS--**
140 HGR
150 HCOLOR = 6:REM SET COLOR TO BLUE
160:
170:
500 REM **--DRAW CAPITAL LETTER I--**
510 HPLOT 20,5 TO 20,35::REM VERTICAL LINE
520 HPLOT 15,5 TO 25,5:::REM HORIZONTAL LINE
530 HPLOT 15,35 TO 25,35:REM HORIZONTAL.LINE
540:
550:
```

This is nearly the same program that draws the high-resolution "I". The only differences are the program-name change in line 100 and the elimination of lines 560 and 570.

The next part of the ANIMATED.I program sets up certain numeric variables that are used in place of the constant values used in lines 510, 520, and 530. REM statements help define the meaning and use of the variables.

```
700 REM **--ANIMATE--**
710 REM CC = CURRENT COLUMN POSITION
720 REM NC = NEW COLUMN POSITION
730 REM CR = CURRENT ROW POSITION
740 REM NR = NEW ROW POSITION
750 REM HZ = HORIZONTAL LENGTH
760 REM VT = VERTICAL LENGTH
770 CC = 20:CR = 5:VT = 30:HZ = INT(VT/6)
780:
```

The length of the horizontal lines is determined by dividing the length of the vertical line by six. The resulting value is one half of the actual length of each horizontal line. The complete horizontal line can be drawn by

1. Subtracting the HZ value from the current column position to determine the starting point of the horizontal line, (CC − HZ)
2. Adding the HZ value to the current column position to determine the ending point of the horizontal line (CC + HZ)
3. A horizontal line can then be drawn at any position on the current row with the instruction:

```
HPLOT CC - HZ,CR TO CC + HZ,CR.
```

The next section of instructions uses the horizontal line formula to erase the capital letter "I" from its current position.

```
790 FOR K = 1 TO 60
800:
810 REM *--ERASE FROM CURRENT POSITION--*
820 HCOLOR = O:REM CHANGE COLOR TO BLACK
830 HPLOT CC,CR TO CC,CR + VT:REM VERT.LINE
840 HPLOT CC - HZ,CR TO CC + HZ,CR:REM TOP LINE
850 HPLOT CC - HZ,CR + VT TO CC + HZ,CR + VT:REM BOT.LINE
860:
```

The FOR-NEXT loop establishes that 60 different capital letter "I"s will be drawn. After changing the color to black to match the background color, the computer is instructed to replot the "I" in its current position. Plotting over an object with the background color is one way of erasing that object from the screen.

The vertical line is plotted by using the value of the length of the "I" and adding that length to the value of the top of the I (CR + VT). As long as the top row of the "I" is known, the coordinates of the vertical line can be determined (CC,CR TO CC,CR + VT) (Fig. 7.9).

The bottom horizontal line of the"I" is the most complicated. You must perform both the calculations involving HZ (CC − HZ, CC + HZ) and the calculation involving VT (CR + VT) in order to determine the coordinates of the bottom horizontal line (CC − HZ,CR + VT TO CC + HZ,CR + VT). Remember that VT must be added to both the starting point and the ending point since this is the horizontal line that is the length VT from the top of the "I".

After the "I" in the current position has been erased you can draw the "I" in its new position.

```
870 REM*--DRAW IN NEW POSITION--*
880 HCOLOR = 6:REM SET COLOR TO BLUE
890 NC = CC + 2:NR = CR + 2
900 HPLOT NC,NR TO NC,NR + VT:REM VERT.LINE
910 HPLOT NC - HZ,NR TO NC + HZ,NR:REM TOP LINE
920 HPLOT NC - HZ,NR + VT TO NC + HZ,NR + VT:REM BOT.LINE
930:
```

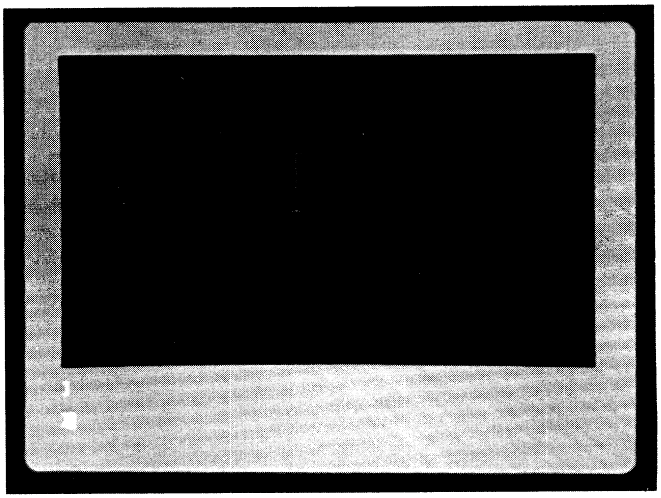


FIGURE 7.9. ANIMATED.I program.

You must reset the color before re-creating the object or you will continue to draw in the color of the background. The new column and row positions are established in this case by adding two to the value of the old positions (NC = CC + 2:NR = CR + 2). The capital letter "I" is then created in the new position.

```
940 CC = NC:CR = NR:REM RESET CURRENT POSITIONS

950 NEXT K

960:

970:
```

After the "I" has been created in its new position, the variables CC and CR must be reset to equal the current position of the "I" (CC = NC:CR = NR). Once these variables have been reset, the computer is told to do the process over again (NEXT K).

The final section of this program ends the animation by displaying the final position of the "I" on the screen until the user presses the RETURN key (INPUT "";L$). At that point, the computer switches from high-resolution graphics back to text and ends the program.

```
5000 REM **--END--

5010 INPUT "";L$

5020 TEXT

5030 END
```

Although this program does not do very much, it does demonstrate in an understandable way the processes that are involved in creating the illusion of movement on the screen. Once the concept of animation is clearly understood (i.e., draw, erase, re-draw), creating programs that involve moving objects is not as difficult as you may have thought.

Clearly there are limitations to this type of animation. If you run the ANIMATED.I program a number of times you will see how much flicker there is and how jerky the image appears as it drops to the bottom of the screen. Certain tricks can be used to reduce both of these limitations. The next section of this chapter discusses these tricks, but please remember that this is an introductory book and not a book designed to instruct readers on how to become graphic artists or arcade-quality programmers.

Page Flipping

Although there are two high-resolution pages, we have so far only used High-Resolution Page 1. One of the tricks programmers use to reduce flickering in animation is to use both high-resolution pages. The process involves drawing, erasing, and redrawing on both pages instead of just one.

Apple provides a way to display one high-resolution page while the computer is creating a drawing on the other high-resolution page. When the drawing on the other page is completed, the computer can be instructed to switch displays and show the completed drawing. While the new display is on the screen, the computer can be working on the original high-resolution page, creating another new drawing. The process of displaying one page while working on the other page and then switching displays is one of the most common methods programmers use to significantly reduce flicker in BASIC animation. To demonstrate this procedure, we will modify the ANIMATED.I program to incorporate page flipping. Load the program back into the computer and make the following changes:

Old line: 100 REM ***--ANIMATED.I--***
New line: 100 REM ***--PAGE.FLIP.I--***

Old line: 140 HGR
New Line: 140 HGR2:HGR:POKE − 16302,0:REM FULL SCREEN

Old line: 790 FOR K = 1 TO 60
New line: 790 FOR K = 1 TO 30

Old line: 810 REM *--ERASE FROM CURRENT POSITION--*
New line: 810 REM *--ERASE FROM PAGE 2--*

Old line: 870 REM *--DRAW IN NEW POSITION--*
New line: 870 REM *--DRAW ON PAGE 2--*

Old line: 890 NC = CC + 2:NR = CR + 2
New line: 890 C2 = C1 + 2:R2 = R1 + 2

Old line: 900 HPLOT NC,NR TO NC,NR + VT:REM VERT.LINE
New line: 900 HPLOT C2,R2 TO C2,R2 + VT:REM VERT.LINE

Old line: 910 HPLOT NC − HZ,NR TO NC + HZ,NR:REM TOP LINE
New line: 910 HPLOT C2 − HZ,R2 TO C2 + HZ,R2:REM TOP LINE

Old line: 920 HPLOT NC − HZ,NR + VT TO NC + HZ, NR + VT:REM BOT.LINE
New line: 920 HPLOT C2 − HZ,R2 + VT TO C2 + HZ,R2 + VT:REM BOT.LINE

Old line: 940 CC = NC:CR = NR:REM RESET CURRENT POSITIONS
New line: 940 CC = C1:CR = R1:REM RESET CURRENT POSITIONS

Delete the following line:

```
950 NEXT K
```

Add the following lines:

```
762 REM C1 = COLUMN POSITION ON PAGE 1
763 REM R1 = ROW POSITION ON PAGE 1
764 REM C2 = COLUMN POSITION ON PAGE 2
765 REM R2 = ROW POSITION ON PAGE 2
775 C1 = 20:R1 = 5
815 POKE 230,64:REM DRAW ON PAGE 2
925 POKE - 16299,0:REM SWITCH DISPLAY TO PAGE 2
1000 REM *--ERASE FROM PAGE 1--*
1010 POKE 230,32:REM DRAW ON PAGE 1
1020 HCOLOR = O:REM CHANGE COLOR TO BLACK
1030 HPLOT CC,CR TO CC,CR + VT:REM VERT.LINE
1040 HPLOT CC - HZ,CR TO CC + HZ,CR:REM TOP LINE
1050 HPLOT CC - HZ,CR + VT TO CC + HZ,CR + VT:REM BOT.LINE
1060:
1070 REM *--DRAW ON PAGE 1--*
1080 HCOLOR = 6:REM SET COLOR TO BLUE
1090 C1 = C2 + 2:R1 = R1 + 2
1100 HPLOT C1,R1 TO C1,R1 + VT:REM VERT.LINE
1110 HPLOT C1 - HZ,R1 TO C1 + HZ,R1:REM TOP LINE
1120 HPLOT C1 - HZ,R1 + VT TO C1 + HZ,R1 + VT:REM BOT.LINE
1130 POKE - 16300,0:REM SWITCH DISPLAY TO PAGE 1
1140:
1150 CC = C1:CR = R2:REM RESET CURRENT POSITIONS
1160 NEXT K
1170:
1180:
```

By making a few changes and adding these 26 lines, you will be able to significantly reduce the amount of flicker you see when the letter "I" moves on a diagonal from near the top of the screen to near the bottom of the screen.

The flow of the program now looks like this:

Display High-Resolution Page 1 (instruction 140)
Draw a capital letter "I" on Page 1 (510–530)
Begin the FOR-NEXT loop (790)
Set the switch so the computer draws on Page 2 (815)
Set the color equal to the background (820)
Draw (erase) a black "I" on page 2 (830–850)
Set the color to blue (880)
Set the column and row values on Page 2 equal to the column and row values on Page 1 plus 2 (890)
Draw a blue "I" on Page 2 (900–920)
Display the "I" on Page 2 (925)
Set the current column and row values equal to the column and row values on Page 1 (940)
Set the switch so the computer draws on Page 1 (1010)
Set the color equal to the background (1020)
Draw (erase) a black "I" on Page 1 (1030–1050)
Set the color to blue (1080)
Set the column and row values on Page 1 equal to the column and row values on Page 2 plus 2 (1090)
Draw a blue "I" on Page 1 (1100–1120)
Display the "I" on Page 1 (1130)
Set the current column and row values equal to the column and row values on Page 2(1150)
Repeat the process 29 more times (1160)
End the program (5000–5030)

The procedure outlined above can be used for any animation that you want to do in BASIC. The only difficult part of the process is remembering that each time you erase, you are not erasing the last object drawn. Instead, you are actually erasing the object drawn before that, because the object is on a different screen from the last object drawn. In other words, the process is: draw on 1, erase on 2, draw on 2, erase on 1, draw on 1, erase on 2, etc. The most difficult part of keeping this straight comes in figuring out the coordinates for the object to be erased. That is why I used different column and row variables for each screen.

At the end of this chapter, I have included a more elaborate demonstration program. I modified the low-resolution MOVING.I program from Chapter 6 so that it works in high-resolution graphics and makes use of page flipping. The "I" can be moved all over the screen with the arrow keys, game paddles, or a joystick. The "I" can also be expanded or contracted using either the Apple keys or paddle or joystick buttons. Because both high-resolution pages are used, the amount of flickering is significantly reduced. If you are

interested in creating animation, I suggest you spend some time studying this NO.FLICKER.I program until you understand how the program uses flags to determine which page to erase or draw on. This program also demonstrates how low-resolution programs can be converted to high-resolution graphics.

COMPILED BASIC

Another trick that is sometimes used to speed up BASIC animation programs is to compile the programs into object code. In order to do this you must have access to a commercial Applesoft BASIC compiler. There are a number of these compilers available, and generally they do speed up the operation of any BASIC program, especially graphic programs. I used a compiler on the NO.FLICKER.I program and then ran it on an Apple IIGS computer in the fast mode. Using a joystick, the response of the "I" to the movement of the joystick was nearly instantaneous, and certainly within the range of some of the commercial game programs available for the Apple.

MEMORY ORGANIZATION

I have tried to show that it is possible to do quite a bit in high-resolution graphics, including page-flipping animation, without the necessity of learning assembly language or other topics that require a certain degree of knowledge about the operation of the computer (such as shapes and shape-table manipulation). But at some point, anyone interested in extensive graphic creation will need to learn about memory organization.

Understanding how the Apple *II* computer uses memory is important because the high-resolution graphic pages occur in the middle of the memory space allocated for Applesoft BASIC programs. In other words, if you have a large BASIC program that uses graphics, it is very possible to wipe out some of the program when it is run and the high-resolution memory is used to display graphics.

Apple *II* computers organize memory into sections that are designed for specific tasks. If memory is not being used for its intended task, that memory can be used for other tasks. But if the intended task is used (called upon), it has priority and will replace anything that is currently there. Applesoft BASIC programs are designed to begin at memory location 800 hex ($800) or 2048 decimal. In other words, all BASIC instructions begin 2048 spaces into the computer's memory. When a programmer adds to the BASIC program, more of the memory is used up. The high-resolution graphics areas begin at memory location 8096 ($2000), which means that a programmer has about 6000 characters of program instruction available before running into the high-resolution-graphic areas of the computer's memory. That is really not very much room for a program. Many programs will fit in that space because the BASIC reserved words (such as PRINT) are "tokenized" or coded so that

instead of taking up the same amount of space as their number of letters, they usually take up only one or two spaces. For example, instead of using five spaces for the BASIC word PRINT, the computer needs only one space. Therefore, you can actually get quite a large program to fit within the space between memory location 2048 (the beginning of Applesoft BASIC programs) and memory location 8096 (the beginning of High-Resolution Page 1).

Eventually, though, you may have a program that will not fit within the space between the start of program memory and the start of high-resolution memory. There are a variety of solutions to this problem. If you have an extremely large program, you may have to split the program so that part of it fits into memory between 2048 and 8096 and the rest goes after the end of high-resolution memory (24576). But if your program is less than approximately 14,000 characters, all you need to do is to place the program after High-Resolution Page 2.

Memory locations 103 and 104 contain the address of the start-of-program memory. By POKEing those locations with a new address, we can move the BASIC program to the new location—above High-Resolution Page 2.

```
POKE 103,1

POKE 104,96
```

The first space of program memory must contain a zero or beginning-of-program code, so one additional instruction is necessary.

```
POKE 24576,0
```

With these three instructions, BASIC programs larger than 6,000 characters can still make use of both high-resolution pages.

LINES AND COLORS

High-resolution graphics is supposed to have an eight-color display on screens measuring 280 by 192. If you watched the creation of the high-resolution rectangle in the LOW HIGH comparison system program (listed in Chapter 5) you should have seen that not every color will appear on every line. The explanation for this phenomenon is complicated, but a simple understanding of the fact that the phenomenon exists is all that you really need. Generally, even-numbered colors can be displayed on even-numbered columns (lines) and odd-numbered colors can be displayed on odd-numbered columns. A short tutorial will quickly demonstrate the situation. (Televisions, composite color monitors, and monochrome monitors may not produce these effects, which are clearly visible on RGB monitors.) Type the following:

```
HGR {RETURN}
```

```
HCOLOR = 5 {RETURN}

HPLOT 101,50 to 101,125 {RETURN}
```

You should see a brown line on the screen. You have used an odd-numbered color (5) on an odd-numbered column (101). Now type

```
CALL-3086 {RETURN}
```

to clear the high resolution screen and then type

```
HPLOT 100,50 to 100,125 {RETURN}
```

This time you should not see a line because you have used an odd-numbered color (5) on an even-numbered column (100). Next, type

```
CALL -3086 {RETURN}
```

again to clear the high-resolution screen and then type

```
HCOLOR = 6 {RETURN}

HPLOT 100,50 TO 100,125 {RETURN}
```

This time a blue line should appear, since you are using an even-numbered color (6) on an even-numbered column (100). But if you change the column to an odd number, you should not be able to see the blue line. Type

```
CALL -3086 {RETURN}

HPLOT 101,50 TO 101,125 {RETURN}
```

The even-numbered color (6) does not appear on the odd-numbered column (101).

In addition to the odd/even problem, some colors will affect other colors if they are next to each other. Type

```
CALL - 3086 {RETURN}

HCOLOR = 6 {RETURN}

HPLOT 100,50 TO 100,125 {RETURN}
```

A blue line should appear. Next, change the color to brown (5), and plot a line in the very next column (101). You should get a brown line next to the blue line. Type

```
HCOLOR = 5 {RETURN}

HPLOT 101,50 TO 101,125 {RETURN}
```

Instead of two lines, one blue and one brown, you have one white line. This demonstrates the problem of adjacent colors affecting one another. For further proof add the following:

```
HCOLOR = 1 {RETURN}

HPLOT 102,50 TO 102,125 {RETURN}
```

Now you have an even larger white line. Although you used an odd-numbered color (1) in an even-numbered column (102), you clearly added to the thickness of the white line. Therefore, it is possible to use odd-numbered colors in even-numbered columns or even-numbered colors in odd-numbered columns, but you may not get what you expect. You must be careful in planning your use of color on the high-resolution screens and understand that not every color can be placed in every location with the assurance that the intended color will appear.

SAVING AND LOADING HIGH-RESOLUTION SCREEN INFORMATION

Storage and retrieval of material on the high-resolution screens is not difficult. In order to save the information on a high-resolution screen you simply need to identify the memory location of the start of the high-resolution page—either $2000 (8192 decimal) for Page 1 or $4000 (16384 decimal) for Page 2—and the amount of memory that is to be saved ($1FFF [8191 decimal] in either case). Both figures, the starting location and the amount of memory to be saved, are then used with the BSAVE command:

```
BSAVE PIC.OF.DOG,A$2000,L$1FFF
```

The above command stores the contents of High-Resolution Page 1 to a file on a diskette called PIC.OF.DOG. The following command does the exact same thing but uses decimal values instead of hexadecimal values:

```
BSAVE PIC.OF.DOG,A8192,L8191
```

The following commands store information from High-Resolution Page 2 in hexadecimal and decimal notation:

```
BSAVE PIC.OF.CAT,A$4000,L$1FFF
```

```
BSAVE PIC.OF.CAT,A16384,L8191
```

Retrieving graphic information from a diskette is easy also. The BLOAD command is used to bring information back from storage on a diskette. In order to see a display of the graphic information you must place the computer in one of its high-resolution graphic modes—HGR, HGR2, or use one of the POKE or CALL switches.

If you want to display the graphic information on the same page from which it was saved, you simply BLOAD the file name.

```
BLOAD PIC.OF.DOG
```

This command displays the graphic information on High-Resolution Page 1 since this information was saved from High-Resolution Page 1

```
BLOAD PIC.OF.CAT
```

displays the graphic information on High-Resolution Page 2 since this information was saved from High-Resolution Page 2.

If you want to display information on a different page than the one from which it was saved, simply supply the computer with the correct memory address for the high-resolution page on which you wish the graphic information displayed. You do not need to indicate the amount (or length) of the information.

```
BLOAD PIC.OF.DOG,A$4000
```

or

```
BLOAD PIC.OF.DOG,A16384
```

will load the graphic information stored in the file called PIC.OF.DOG and display that information on High-Resolution Page 2 even though the information originally came from High-Resolution Page 1.

```
BLOAD PIC.OF.CAT,A$2000
```

or

```
BLOAD PIC.OF.CAT,A8192
```

will load the graphic information stored in the file called PIC.OF.CAT and display that information on High-Resolution Page 1 even though the information originally came from High-Resolution Page 2.

If you are using these commands from within an Applesoft BASIC program (under either DOS 3.3 or ProDOS), they must be preceded with a CHR$(4) (the ASCII value for control D). To accomplish this, these commands are included in a PRINT statement.

```
PRINT CHR$(4); "BLOAD PIC.OF.DOG"
```

or

```
PRINT CHR$(4); "BSAVE PIC.OF.CAT, A$4000, L$1FFF"
```

Often programmers will set the variable D$ equal to CHR$(4) and then use that value in all PRINT statements referring to the disk drive:

```
D$ = CHR$(4)
PRINT D$; "BSAVE PIC.OF.CAT,A16384,L8191"
```

or

```
PRINT D$; "BLOAD PIC.OF.CAT"
```

Storage of high-resolution information in this manner is expensive in terms of the space taken up on a diskette. Under DOS 3.3, storage of each high-resolution page uses 34 sectors of space. Storage and retrieval of that amount of space is also not very fast. Other methods of storing and retrieving high-resolution information are available, but those methods require considerably more programming and a knowledge of screen-display memory. If you need to store a great deal of graphic information and/or recall information relatively quickly, then you should probably learn some of the tricks of saving space in storage of high-resolution information. For most people, the method of storage and retrieval described here should prove sufficient

In the next chapter, I will discuss the method used to display text on either of the high-resolution screens. This method is also used to create many arcade games and other displays that require quickly drawn objects.

HIGH-RESOLUTION PEEKS, POKES, AND CALLS

POKE −16297,0 Switches the display from low-resolution to high-resolution if the display is in the graphic mode. If the display is in the text mode, POKE − 16297,0 will not affect the display.

POKE −16298,0 Switches the display from high-resolution to low-resolution if the display is in the graphic mode. If the display is in the text mode, POKE −16298,0 will not affect the display.

POKE −16299,0 Sets the display to Page 2. If the display is in the high-resolution graphic mode, POKE −16299,0 will switch the display from High-Resolution Page 1 to High-Resolution Page 2. Care must be used with this switch, since the area of memory assigned to Text Page 2 is also used for storage of Applesoft BASIC programs. If the display is in the text mode, POKE −16299,0 will switch to Text Page 2 and show inverse characters that are part of the BASIC program.

POKE −16300,0 Sets the display to Page 1. If the display is in the high-resolution graphic mode, POKE −16300,0 will switch the display from High-Resolution Page 2 to High-Resolution Page 1.

POKE −16301,0 Sets the display to both graphics and text. This switch only works after the graphics mode has been set (it has no effect when the display is in the text mode). Care must be used with this switch. The text part of this switch comes from the page being displayed. If graphic (either high- or low-resolution) Page 1 is displayed, POKE −16301,0 will display graphic and text information from Page 1. If graphic Page 2 is displayed, POKE −16301,0 will display graphic and text information from Page 2 (with the text information composed of inverse characters coming from the Applesoft BASIC program).

POKE −16302,0 Sets the display to full-screen graphics. If the display is in the text mode, this switch has no effect on the display.

POKE −16303,0 Sets the display to full-screen text. If the display is in the text mode, this switch has no effect on the display. If the display is in any graphic mode, the computer switches to the text mode. This POKE is similar to the TEXT command, but does not reset the screen default values.

POKE −16304,0 Sets the display to some type of graphic mode, depending upon how other switches are set. If no other switches have previously been set, the graphic display comes from Page 1, with both text and graphics visible.

POKE 230,32 Instructs the computer to perform high-resolution commands on High-Resolution Page 1 regardless of which screen is currently being displayed. In computer jargon, the "pen" is set to Page 1.

POKE 230,64 The pen is set to Page 2 no matter what the current display is. After this POKE, the computer will do all graphic work on Page 2 until specifically instructed not to.

CALL −3086 Clears the current high-resolution page to black.

CALL −3082 Clears the current high-resolution page to the last color used in an HPLOT statement.

PEEK (234) Collision Counter. This memory location can be used with SHAPE commands to determine when two or more objects occupy the same location on the screen at the same time (more about this in the next chapter).

REVIEW QUESTIONS

1. What high-resolution BASIC command is used to draw a horizontal line?
2. Which value, column or row, is placed first in the definition of a specific point on the high-resolution screens?
3. What are the screen coordinates for the upper left-hand corner on either high-resolution screen?
4. What are the screen coordinates for the lower-right-hand corner on either high-resolution screen?
5. True or False? In order to instruct the computer to draw a circle, you must understand certain trig functions.
6. True or False? It is not possible to change the background color of the high-resolution screens.
7. What are the three basic operations in computer animation?
8. What memory location (decimal value) is used to inform the computer which high-resolution page to draw on?
9. Approximately how many characters of BASIC programming can fit between the start of program memory and the start of high-resolution page one?
10. What command is used to save high-resolution screen memory?

High-Res "I"

```
100  REM   ***--HI.RES.I--***
110 :
120 :
130  REM   **--INITIALIZE HI-RES GRAPHICS--**
140  HGR
150  HCOLOR= 6: REM   SET COLOR TO BLUE
160 :
170 :
500  REM   **--DRAW CAPITAL LETTER I--**
510  HPLOT 20,5 TO 20,35:: REM  VERTICAL LINE
520  HPLOT 15,5 TO 25,5::: REM  HORIZONTAL LINE
530  HPLOT 15,35 TO 25,35: REM  HORIZONTAL LINE
540 :
550 :
560  REM   **--END PROGRAM--**
570  END
```

High-Res Circle

```
100  REM   ***--HI.RES.CIRCLE--***
110 :
120 :
130  REM   **--INITIAL VALUES--**
140 PI = 3.1415926:::: REM  VALUE FOR PI
150 COLUMN1 = 140::::: REM  FIRST COLUMN POSITION
160 ROW1 = 80::::::::: REM  FIRST ROW POSITION
170 RADIUS = 70::::::: REM  SIZE OF CIRCLE
180 NUMBER = 256:::::: REM  NUMBER OF POINTS ON CIRCLE
190 CC = 2 * PI::::::: REM  EXTENT OF LOOP
200 INC = CC / NUMBER: REM  INCREMENT IN LOOP
210 :
220 :
500  REM   **--SET UP GRAPHICS--**
510  HGR
520  HCOLOR= 3: REM  USE WHITE TO DISPLAY ALL PTS
530 :
540 :
700  REM   **--DRAW CIRCLE--**
710  FOR I = 0 TO CC STEP INC
720 CLMN = RADIUS *  SIN (I)
730 ::RW = RADIUS *  COS (I)
735 RW = RW * .83: REM  ADJUST FOR ELLIPSE EFFECT
740  HPLOT COLUMN1 + CLMN,ROW1 + RW
750  NEXT I
```

```
760 :
770 :
800  REM  **--END--**
810  END
```

Background Demo

```
100  REM  ***--BACKGROUND.DEMO--***
110 :
120 :
130  REM  **--INITIALIZE GRAPHICS--**
140  HGR
160 :
170 :
200  REM  **--CHANGE BACKGROUND--**
210  FOR COLR = 0 TO 7
220  HCOLOR= COLR
230  HPLOT 0,0
240  CALL  - 3082
250  INPUT "";L$: REM  WAIT UNTIL USER READY TO CONT.
260  NEXT COLR
270 :
280 :
290  REM  **--END--**
300  TEXT
310  END
```

Animated "I"

```
100  REM  ***--ANIMATED.I--***
110 :
120 :
130  REM   **--INITIALIZE HI-RES GRAPHICS--**
140  HGR
150  HCOLOR= 6: REM   SET COLOR TO BLUE
160 :
170 :
500  REM  **--DRAW CAPITAL LETTER I--**
510  HPLOT 20,5 TO 20,35:: REM  VERTICAL LINE
520  HPLOT 15,5 TO 25,5::: REM  HORIZONTAL LINE
530  HPLOT 15,35 TO 25,35: REM  HORIZONTAL LINE
540 :
550 :
700  REM  **--ANIMATE--**
710  REM  CC = CURRENT COLUMN POSITION
```

```
720  REM  NC = NEW COLUMN POSITION
730  REM  CR = CURRENT ROW POSITION
740  REM  NR = NEW ROW POSITION
750  REM  HZ = HORIZONTAL LENGTH
760  REM  VT = VERTICAL LENGTH
770 CC = 20:CR = 5:VT = 30:HZ =  INT (VT / 6)
780 :
790  FOR K = 1 TO 60
800 :
810  REM  *--ERASE FROM CURRENT POSITION--*
820  HCOLOR= 0: REM  CHANGE COLOR TO BLACK
830  HPLOT CC,CR TO CC,CR + VT: REM  VERT.LINE
840  HPLOT CC - HZ,CR TO CC + HZ,CR: REM  TOP LINE
850  HPLOT CC - HZ,CR + VT TO CC + HZ,CR + VT: REM  BOT.LINE
860 :
870  REM  *--DRAW IN NEW POSITION--*
880  HCOLOR= 6: REM  SET COLOR TO BLUE
890 NC = CC + 2:NR = CR + 2
900  HPLOT NC,NR TO NC,NR + VT: REM  VERT.LINE
910  HPLOT NC - NZ,NR TO NC + HZ,NR: REM  TOP LINE
920  HPLOT NC - HZ,NR + VT TO NC + HZ,NR + VT: REM  BOT.LINE
930 :
940 CC = NC:CR = NR: REM  RESET CURRENT POSITIONS
950  NEXT K
960 :
970 :
5000  REM  **--END--**
5010  INPUT "";L$
5020  TEXT
5030  END
```

Page Flip

```
100  REM  ***--PAGE.FLIP.I--***
110 :
120 :
130  REM   **--INITIALIZE HI-RES GRAPHICS--**
140  HGR2 : HGR : POKE  - 16302,0: REM  FULL SCREEN
150  HCOLOR= 6: REM   SET COLOR TO BLUE
160 :
170 :
500  REM  **--DRAW CAPITAL LETTER I--**
510  HPLOT 20,5 TO 20,35:: REM  VERTICAL LINE
520  HPLOT 15,5 TO 25,5::: REM  HORIZONTAL LINE
530  HPLOT 15,35 TO 25,35: REM  HORIZONTAL LINE
540 :
```

```
550 :
700  REM  **--ANIMATE--**
710  REM  CC = CURRENT COLUMN POSITION
720  REM  NC = NEW COLUMN POSITION
730  REM  CR = CURRENT ROW POSITION
740  REM  NR = NEW ROW POSITION
750  REM  HZ = HORIZONTAL LENGTH
760  REM  VT = VERTICAL LENGTH
762  REM  C1 = COLUMN POSITION ON PAGE 1
763  REM  R1 = ROW POSITION ON PAGE 1
764  REM  C2 = COLUMN POSITION ON PAGE 2
765  REM  R2 = ROW POSITION ON PAGE 2
770 CC = 20:CR = 5:VT = 30:HZ =  INT (VT / 6)
775 C1 = 20:R1 = 5
780 :
790  FOR K = 1 TO 30
800 :
810  REM  *--ERASE FROM PAGE 2--*
815  POKE 230,64: REM  DRAW ON PAGE 2
820  HCOLOR= 0: REM  CHANGE COLOR TO BLACK
830  HPLOT CC,CR TO CC,CR + VT: REM  VERT.LINE
840  HPLOT CC - HZ,CR TO CC + HZ,CR: REM  TOP LINE
850  HPLOT CC - HZ,CR + VT TO CC + HZ,CR + VT: REM  BOT.LINE
860 :
870  REM  *--DRAW ON PAGE 2--*
880  HCOLOR= 6: REM   SET COLOR TO BLUE
890 C2 = C1 + 2:R2 = R1 + 2
900  HPLOT C2,R2 TO C2,R2 + VT: REM   VERT.LINE
910  HPLOT C2 - HZ,R2 TO C2 + HZ,R2: REM   TOP LINE
920  HPLOT C2 - HZ,R2 + VT TO C2 + HZ,R2 + VT: REM   BOT.LINE
925  POKE  - 16299,0: REM  SWITCH DISPLAY TO PAGE 2
930 :
940 CC = C1:CR = R1: REM   RESET CURRENT POSITIONS
960 :
970 :
1000  REM   *--ERASE FROM PAGE 1--*
1010  POKE 230,32: REM   DRAW ON PAGE 1
1020  HCOLOR= 0: REM   CHANGE COLOR TO BLACK
1030  HPLOT CC,CR TO CC,CR + VT: REM   VERT.LINE
1040  HPLOT CC - HZ,CR TO CC + HZ,CR: REM   TOP LINE
1050  HPLOT CC - HZ,CR + VT TO CC + HZ,CR + VT: REM   BOT.LINE
1060 :
1070  REM   *--DRAW ON PAGE 1--*
1080  HCOLOR= 6: REM   SET COLOR TO BLUE
1090 C1 = C2 + 2:R1 = R2 + 2
1100  HPLOT C1,R1 TO C1,R1 + VT: REM    VERT.LINE
1110  HPLOT C1 - HZ,R1 TO C1 + HZ,R1: REM    TOP LINE
1120  HPLOT C1 - HZ,R1 + VT TO C1 + HZ,R1 + VT: REM    BOT.LINE
```

```
1130  POKE  - 16300,0: REM    SWITCH DISPLAY TO PAGE 1
1140 :
1150 CC = C2:CR = R2: REM    RESET CURRENT POSITIONS
1160  NEXT K
1170 :
1180 :
5000  REM  **--END--**
5010  INPUT "";L$
5020  TEXT
5030  END
```

No-Flicker "I"

```
100  REM ***--NO.FLICKER.I--***
110 :
120 :
130  REM  **--INITIALIZE HI-RES GRAPHICS--**
140  INPUT "WHICH METHOD (K/P/J) ";METHOD$
150  HGR2 : HGR :FLAG$ = "2": POKE  - 16302,0: REM  FULL SCREEN
160  HCOLOR= 6: REM SET COLOR TO BLUE
170 :
180  REM  *--INITIAL VALUES--*
190 VERTLGTH = 30: REM  VERTICAL SIZE
200 HZLGTH =  INT (VERTLGTH / 6): REM  HORIZONTAL SIZE
210 X = 20:Y = 5: REM  BEGINNING LOCATION
220 :
230  REM  *--X2,Y2,V2,AND H2 ARE VALUES TO BE ERASED--*
240  REM  *--ON SCREEN NOT BEING DISPLAYED--*
250 X2 = X:Y2 = Y:X1 = X:Y1 = Y
260 V2 = VERTLGTH:H2 = HZLGTH:V1 = VERTLGTH:H1 = HZLGTH
270 :
280  REM  *--INITIALIZE KEYS--*
290 KEY =  PEEK ( - 16384)
300 SA =  PEEK ( - 16286): REM  STORE VALUE OF BUTTON 1/SOLID APPLE KEY
310 OA =  PEEK ( - 16287): REM  STORE VALUE OF BUTTON 0/OPEN APPLE KEY
320  IF METHOD$ = "P" OR METHOD$ = "p" OR METHOD$ = "J" OR METHOD$ = "j"
     THEN 2000: REM      SKIP OVER KEYBOARD LINES
330 :
340 :
350  REM  **--USE OF KEYBOARD--**
360  IF (KEY - 128) = 8 THEN X = X - 2: GOTO 550: REM    LEFT ARROW PRES
    SED
370  IF (KEY - 128) = 21 THEN X = X + 2: GOTO 550: REM   RIGHT ARROW PRE
    SSED
380  IF (KEY - 128) = 10 THEN Y = Y + 2: GOTO 550: REM   DOWN ARROW PRES
    SED
```

```
390  IF (KEY - 128) = 11 THEN Y = Y - 2: GOTO 550: REM   UP ARROW PRESSE
   D
400 :
410 :
420  REM   **--ARROW KEYS--**
430  IF SA > 127 THEN VERTLGTH = VERTLGTH - 1:HZLGTH =  INT (VERTLGTH /
   6): GOTO 550: REM SOLID APPLE PRESSED
440  IF OA > 127 THEN VERTLGTH = VERTLGTH + 1:HZLGTH =  INT (VERTLGTH /
   6): GOTO 550: REM OPEN APPLE PRESSED
450  IF (KEY - 128) = 27 THEN 490: REM ESC KEY PRESSED SO END
460  GOTO 290
470 :
480 :
490  REM  **--END PROGRAM--**
500  POKE  - 16368,0: REM  RESET KEYBOARD STROBE
510  TEXT : END
520 :
530 :
540 :
550  REM   **--DRAW CAPITAL LETTER I--**
560  IF FLAG$ = "2" THEN  POKE 230,32: GOSUB 800: GOTO 580: REM  DISPLAY
    PG 2 & DRAW PG 1
570  IF FLAG$ = "1" THEN  POKE 230,64: GOSUB 800: GOTO 580: REM  DISPLAY
    PG 1 & DRAW PG 2
580  HCOLOR= 3
590 :
600  REM  *--CHECK FOR VALUE OUT OF BOUNDS--*
610  IF X - HZLGTH < 0 THEN X = HZLGTH: GOTO 630
620  IF X + HZLGTH > 279 THEN X = 279 - HZLGTH
630  IF Y + VERTLGTH > 191 THEN Y = 191 - VERTLGTH
640  IF Y < 0 THEN Y = 0
650  IF VERTLGTH > 191 THEN VERTLGTH = 191
660  IF VERTLGTH < 0 THEN VERTLGTH = 0
670 :
680  REM  *--PLOT VALUES--*
690  HPLOT X,Y TO X,Y + VERTLGTH: REM VERTICAL LINE
700  HPLOT X - HZLGTH,Y TO X + HZLGTH,Y; REM TOP HORIZ.LINE
710  HPLOT X - HZLGTH,Y + VERTLGTH TO X + HZLGTH,Y + VERTLGTH: REM BOTTO
   M LINE
720 :
730 X2 = X1:Y2 = Y1:X1 = X:Y1 = Y:V2 = V1:H2 = H1:V1 = VERTLGTH:H1 = HZL
   GTH
740  POKE  - 16368,0: REM  RESET KEYBOARD STROB
750  IF FLAG$ = "2" THEN  POKE  - 16300,0:FLAG$ = "1"; GOTO 290: REM  SW
    ITCH DISPLAY TO PG 1
760  IF FLAG$ = "1" THEN  POKE  - 16299,0:FLAG$ = "2": GOTO 290: REM  SW
    ITCH DISPLAY TO PG 2
770  RETURN
```

```
780 :
790 :
800  REM **--ERASE LAST POSITION ON OTHER SCREEN-**
810  HCOLOR= 0
820  HPLOT X2,Y2 TO X2,Y2 + V2: REM VERTICAL LINE
830  HPLOT X2 - H2,Y2 TO X2 + H2,Y2: REM TOP HORIZ.LINE
840  HPLOT X2 - H2,Y2 + V2 TO X2 + H2,Y2 + V2: REM BOTTOM LINE
850  RETURN
860 :
870 :
2000  REM  **--PADDLE POSITION--**
2010 P0 =  PDL (0)
2020 :
2030 P1 =  PDL (1)
2040 :
2050  IF P1 > 127 AND P1 < 132 THEN 2080
2060  IF P1 < 131 THEN Y = Y - 2: REM UP
2070  IF P1 > 131 THEN Y = Y + 2: REM DOWN
2080  IF P0 > 127 AND P0 < 132 THEN 3000
2090  IF P0 > 131 THEN X = X + 2: REM RIGHT
2100  IF P0 < 131 THEN X = X - 2: REM LEFT
2110  IF Y < 0 THEN Y = 0
2120 :
2130 :
3000  REM **--APPLE KEYS--**
3010  IF SA > 127 THEN VERTLGTH = VERTLGTH - 1:HZLGTH =  INT (VERTLGTH /
    6): GOTO 550: REM SOLID APPLE PRESSED
3020  IF OA > 127 THEN VERTLGTH = VERTLGTH + 1:HZLGTH =  INT (VERTLGTH /
    6): GOTO 550: REM OPEN APPLE PRESSED
3030  IF (KEY - 128) = 27 THEN 490: REM ESC KEY PRESSED SO END
3040  GOTO 550
```

Polar Roses

```
100  REM   ***--POLAR.ROSES--***
110 :
120 :
130  REM  **--INITIAL VALUES--**
140 PI = 3.1415926:::: REM  VALUE FOR PI
150 COLUMN1 = 40:::::: REM  FIRST COLUMN POSITION
160 ROW1 = 130:::::::: REM  FIRST ROW POSITION
170 RADIUS = 10::::::: REM  SIZE OF ELLIPSE
180 NUMBER = 80::::::: REM  NUMBER OF POINTS ON ELLIPSE
190 CC = 2 * PI::::::: REM  EXTENT OF LOOP
200 INC = CC / NUMBER: REM  INCREMENT IN LOOP
210 :
```

```
220 :
500  REM  **--SET UP GRAPHICS--**
510  HGR : POKE  - 16302,0: REM   FULL SCREEN
520 CR = 1: REM  FIRST COLOR = GREEN
525 R = 37: REM  SIZE DETERMINER
530 :
540 :
700  REM  **--DRAW POLAR ROSE--**
705  HCOLOR= CR
710  FOR I = 0 TO CC STEP INC
715 RADIUS = R *  COS (2 * I)
720 CLMN = RADIUS *  SIN (I)
730 ::RW = RADIUS *  COS (I)
735 RW = RW * .83: REM  ADJUST FOR ELLIPSE EFFECT
740  HPLOT COLUMN1,ROW1 TO COLUMN1 + CLMN,ROW1 + RW
750  NEXT I
755 ROW1 = ROW1 - 22:COLUMN1 = COLUMN1 + 50:CR = CR + 1
756  IF CR = 4 THEN CR = CR + 1
757  IF CR = 7 THEN 800
759  GOTO 700
760 :
770 :
800  REM  **--END--**
810  INPUT "";L$
820  TEXT
830  END
```

Fill Demo

```
100  REM  ***--FILL.DEMO--***
110 :
120 :
130  REM  **--INITIAL VALUES--**
140 PI = 3.1415926:::: REM  VALUE FOR PI
150 COLUMN1 = 140::::: REM  FIRST COLUMN POSITION
160 ROW1 = 80::::::::: REM  FIRST ROW POSITION
170 RADIUS = 70::::::: REM  SIZE OF CIRCLE
180 NUMBER = 256:::::: REM  NUMBER OF POINTS ON CIRCLE
190 CC = 2 * PI::::::: REM  EXTENT OF LOOP
200 INC = CC / NUMBER: REM  INCREMENT IN LOOP
210 :
220 :
500  REM  **--SET UP GRAPHICS--**
510  HGR
520  HCOLOR= 3: REM  USE WHITE TO DISPLAY ALL PTS
530 :
```

```
540 :
700  REM   **--DRAW CIRCLE--**
710  FOR I = 0 TO CC STEP INC
720 CLMN = RADIUS *  SIN (I)
730 ::RW = RADIUS *  COS (I)
735 RW = RW * .83: REM  ADJUST FOR ELLIPSE EFFECT
740  HPLOT COLUMN1 + CLMN,ROW1 + RW
750  NEXT I
755  GOSUB 5000: REM  PAUSE
760 :
770 :
900  REM   **--FILL CIRCLE--**
905  HCOLOR= 6:NUMBER = 512
907 INC = CC / NUMBER: REM INCREMENT IN LOOP
910  FOR I = 0 TO CC STEP INC
920 CLMN = RADIUS *  SIN (I)
930 ::RW = RADIUS *  COS (I)
935 RW = RW * .83: REM   ADJUST FOR ELLIPSE EFFECT
940  HPLOT COLUMN1,ROW1, TO COLUMN1 + CLMN,ROW1 + RW
950  NEXT I
955  GOSUB 5000: REM  PAUSE
960 :
970 :
1000  REM  **--DRAW RECTANGLE--**
1010  HGR
1020  HCOLOR= 3
1030  REM  *--BEGIN TOP LEFT CORNER--*
1040  HPLOT 50,50 TO 225,50 TO 225,150 TO 50,150 TO 50,50
1050  GOSUB 5000: REM  PAUSE
1060 :
1070 :
1100  REM  **--FILL RECTANGLE--**
1110  HCOLOR= 1
1120  FOR I = 50 TO 225
1130  HPLOT I,50 TO I,150
1140  NEXT I
1150  GOSUB 5000: REM  PAUSE
1160 :
1170 :
2000  REM  **--DRAW TRIANGLE--**
2010  HGR
2020  HCOLOR= 3
2030  REM  *--BEGIN LOWER LEFT CORNER--*
2040  HPLOT 75,125 TO 225,125 TO 150,25 TO 75,125
2050  GOSUB 5000: REM  PAUSE
2060 :
2070 :
2100  REM  **--FILL TRIANGLE--**
```

```
2110  HCOLOR= 5
2120  FOR I = 75 TO 225
2130  HPLOT I,125 TO 150,25
2140  NEXT I
2150  GOSUB 5000: REM  PAUSE
2160 :
2170 :
4000  REM  **--END--**
4010  TEXT
4020  END
4030 :
4040 :
5000  REM   **--PAUSE--**
5010  INPUT "";L$
5020  RETURN
```

CHAPTER 8

High-Resolution Shapes

Programmers may love what they can do with high-resolution shapes, but most hate the process of creating those shapes when they have to do it by hand. The process can be frustrating, tedious, prone to errors and time consuming. Commercial programs have been developed to replace the hand drawing method, and many individuals have created their own shape-table routines. Authors of programming books give lengthy explanations of the shape-drawing process, only to urge their readers to avoid it by using either a commercial program or program that appeared in a magazine or book. Other authors have ignored hand drawing completely or told readers to use a specific program. The point is that no one is quite sure how to write about high-resolution shapes. The topic cannot be ignored because of the usefulness of the shapes and their essential role in producing text on the high-resolution screens.

In keeping with the somewhat different approach to graphics taken in this book, I encourage readers to follow the explanation of the creation of shapes. In spite of the supposed difficulty involved in creating shapes, the truth is that (like most things) when properly explained and understood, shape-table creation is not that hard. It does take some time and certainly provides ample opportunities for making mistakes, but if you are patient and careful, the results are rewarding.

Most authors of high-resolution graphics books use computer programs (either of their own creation or commercial programs) to reduce the possibility of mistakes in the creation of a shape table. Their text is then taken up with explaining the computer program rather than the shape-creation process.

I have tried to address this problem by explaining the shape-creation process and including the SHAPE.MAKER program, which exactly follows my explanation. In other words, you will not waste time by reading my explanation of the shape-creation process, because the program I have included in this chapter operates according to the steps in that process.

SHAPE DEFINITION

Shapes are objects, drawings, pictures, designs, fonts or anything you can produce with a pen or pencil. The capital letter "I" used for demonstrations in earlier chapters is a shape. Since most of the things created as shapes can also be created by plotting individual points and/or lines, why should we go to the trouble of creating shapes when we can do the same things with the HPLOT command?

The answer is in the way the computer handles predefined objects (shapes). If the computer has certain information about the object, it can rapidly calculate new dimensions, reposition, resize, rotate, and replot the object on either high-resolution screen. The key word is "rapidly." Although the HPLOT statement can handle all of these functions, the amount of programming necessary is considerable, and the speed of execution is drastically reduced. Even when compiled, HPLOT instructions cannot compare to the speed of a program using shapes and shape-table commands, especially when the objects involved are complex or when complex operations (such as rotating and scaling) are necessary. In fact, the ability to manipulate shapes in shape tables makes it possible to create screens that would otherwise be too difficult or impractical.

High-resolution text is a good example of this. Theoretically, a set of HPLOT instructions can create alphanumeric characters anywhere on a high-resolution screen, but the amount of code in such a high-resolution character set would be significant to say the least. In addition, the amount of code and necessary calculations would make the program very slow. By contrast, an alphanumeric shape table, once created, can produce text anywhere on either high-resolution screen with a near instantaneous response. In addition, shape-table objects can be enlarged or reduced, rotated, or repositioned with very few BASIC instructions.

Drawing Shapes

The first step in shape creation is to have a shape in mind and to sit down and draw a rough design of the object. For this tutorial I will use a simple horizontal line, and move on to more complex shapes later.

On paper, a line is actually a line, but on the computer screen, a line is a series of adjacent dots. The second step, therefore, is to draw the shape using only dots. At this point, graph paper becomes very helpful, although for a shape as simple as a horizontal line, it is not necessary. But when you get to the creation of more complex shapes, using graph paper makes the job much easier. Using dots our shape now looks like this:

. . . .

The next step is quite important: you must decide where the computer should begin drawing the shape. In the case of this horizontal-line example, this decision may sound trivial, but it is not. The point at which the computer is instructed to begin drawing the shape is the point used to position the shape on the screen. In other words, if you construct your shape by starting at the left end of the line, the computer will draw the shape with the left end at the location specified in the DRAW command. For example, a specification of 100,50 in a DRAW command would position the starting point, or left end, of the line at column 100 on row 50. If, instead of drawing your shape by starting at the left end of the line, you start at the right end, the computer would draw the line with the right end at location 100,50. Therefore, if you construct the line from left to right, the computer will display the line to the right of the point specified in a DRAW command.

```
(100,50)______________
```

If you construct the shape from right to left, the computer will display the line to the left of the point specified in the DRAW command.

```
______________(100,50)
```

If you construct the shape by beginning in the middle, the computer will display the line with the middle point of the line on the location specified in the DRAW command.

```
_______(100,50)_______
```

As you can see, the starting point in the construction of a shape is very important. Generally, it is a good idea to begin the construction of a shape in the approximate middle of the shape, but that rule is very loose and can be ignored in many situations.

In order to make the explanation of shape construction as easy to understand as possible, I will assume that you will begin constructing the horizontal-line shape at the left edge of the line.

Coding the Vectors

From the leftmost dot on your drawing, draw an arrow to the right. From the next dot draw another right arrow and continue until all seven dots have right arrows. These dots and arrows are now called **vectors**. The term vector is very important in shape creation. Vectors are the instructions that indicate where the shape begins and exactly how the computer is to construct the shape from that beginning point. These instructions or vectors must be put in a form that the computer can understand and use. Most of the confusion in creating shapes occurs in translating the shape into vectors.

Applesoft provides a specific method of translating vectors into instructions that the computer can understand. After you read and understand the explanation, you will be able to use the SHAPE.MAKER program to do the translation itself on the computer.

To facilitate the translation process, Apple designed a code of 1's and 0's that can represent eight different vectors. In your horizontal-line shape you will need only one type of vector: i.e., plot and move right. Each dot represents a point to be plotted, and each arrow indicates a move to the right (Fig. 8.1). Therefore, your seven dots with arrows are seven plot-and-move-right vectors. If we had started on the right end of the line, the arrows would have been towards the left and we would have had seven plot-and-move-left vectors. The following are the eight different types of vectors that can be used to create shapes (Fig. 8.2):

Plot and move right
Plot and move left
Plot and move up
Plot and move down
Do not plot but move right
Do not plot but move left
Do not plot but move up
Do not plot but move down

The binary (1's and 0's) code that Apple uses to represent these eight different vectors is listed below:

Plot and move right = 101
Plot and move left = 111
Plot and move up = 100
Plot and move down = 110
Do not plot but move right = 001
Do not plot but move left = 011
Do not plot but move up = 000
Do not plot but move down = 010

The first step in the translation process is to substitute the proper code for each vector in the shape. Apple suggests redrawing each of the vectors in a straight line and then translating the vector into binary code, but you may find that this unwrapping process is not necessary, especially for a simple shape. When you translate the seven plot-and-move-right vectors in your horizontal-line shape, you get the following code:

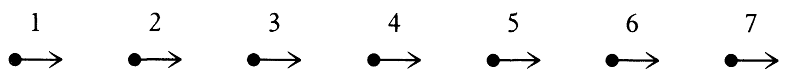


FIGURE 8.1. Horizontal line vectors.

•→	Plot and move right
←•	Plot and move left
↑ (dot below)	Plot and move up
↓ (dot above)	Plot and move down
→	Do not plot but move right
←	Do not plot but move left
↑	Do not plot but move up
↓	Do not plot but move down

FIGURE 8.2. Eight types of vectors.

Vector	7	6	5	4	3	2	1
Binary	101	101	101	101	101	101	101

The next step is to divide this binary code into eight-bit bytes. With the exception of the Apple IIGS, the Apple *II* line of computers operates by processing a maximum of eight pieces (bits) of information at a time (for further explanation, see Chapter 4). Each of the l's or 0's is called a bit, and it takes 8 of these bits to make up one byte of information. Therefore, it is necessary to place the binary code for the horizontal line into groups of eight. Unfortunately, the process is not as simple as just taking eight at a time.

First, each vector's binary code must be assigned to one of three columns. The first vector's code is placed in the right-hand column (A) and the second vector's code is placed in the middle column (B). After the first two vectors, the process becomes more complicated. The third vector's code contains three bits also, and if you add all three of those bits to the six bits you already have from the first two vectors, you will have a total of nine bits of information. But Apple computers can only handle a maximum of eight pieces of information at a time. You must look at the third vector to determine if it contains a zero in its leftmost position. If it does contain a zero, you may be able to drop the zero and place the other two bits in the two available positions of column C, thus completing the eight bits of the first byte. Any vector containing a zero as its leftmost bit can go into column C as long as one of the two remaining bits is a one. In other words, all three bits may not be zeros, because column C cannot contain a vector that is all zeros. The reason for this is simple. Zeros are added to the two positions available in column C whenever the next vector will

not fit into column C, therefore, the computer would not be able to distinguish between zeros added to fill space and zeros included to indicate a do-not-plot-but-move-up vector (000).

If the third vector does not contain a zero in the leftmost bit position, the third vector must be placed into column A (in other words, the third vector must begin a new byte) and zeros are added to the two available positions of column C to fill up the first byte of vector information. Each byte can therefore contain the binary codes for two or three vectors, depending upon what those vectors are, with one exception. If a do-not-plot-but-move-up vector (000) appears in column B and the following vector will not fit into column C, then the do-not-plot-but-move-up vector must be moved from column B to column A in the next byte. Once again, the computer would interpret the zeros as fillers instead of as a vector. For this same reason, you cannot have two do-not-plot-but-move-up vectors in a row.

Each byte of vector information is therefore composed of three binary codes in column A, three binary codes in column B and either two binary codes in column C or two filler zeros in column C.

For your horizontal-line shape, the placement of the vector binary codes is very easy. All seven codes are the same, and none of the codes contain a zero in the leftmost position. Therefore, you cannot place any of the vector codes in column C and, as a result, each byte of vector information will contain two sets of binary codes and two zeros, except the last byte, which will contain one set of binary codes in column A and filler zeros in columns B and C. You will note that the columns below run from right to left; this is because bytes are numbered from right to left (76543210).

	C	*B*	*A*
Vector		2	1
	00	101	101
		4	3
	00	101	101
		6	5
	00	101	101
			7
	00	000	101

The shape is completed by adding an additional byte containing nothing but zeros in all three columns.

00	000	000

Hexadecimal Values

Once each byte of vector codes contains eight bits of information (either codes or filler zeros) we can translate the contents of the bytes into a language the machine can understand

(machine language or hexadecimal notation). Originally, computers were programmed using binary codes—1's and 0's or on/off switches—but today (generally), only the lowest level of programming uses hexadecimal notation or machine language. Therefore, we must take the eight bits of binary code in each byte of vector information and translate the binary code into hexadecimal values. To do this we divide the eight bits into two groups of four bits each and translate each group according to the following table:

BINARY		*HEXADECIMAL*
0000	=	0
0001	=	1
0010	=	2
0011	=	3
0100	=	4
0101	=	5
0110	=	6
0111	=	7
1000	=	8
1001	=	9
1010	=	A
1011	=	B
1100	=	C
1101	=	D
1110	=	E
1111	=	F

Therefore, we take our five bytes, each containing eight bits of binary code, and translate them into hexadecimal values:

Byte 1	00 101 101	00101101
Byte 2	00 101 101	00101101
Byte 3	00 101 101	00101101
Byte 4	00 000 101	00000101
Byte 5	00 000 000	00000000

These five bytes now become:

Left 4 Bits	*Right 4 Bits*
0010	1101
0010	1101
0010	1101
0000	0101
0000	0000

When translated into hexadecimal values, using the table, you get:

Left 4 Bits	*Right 4 Bits*	*Hex Values For Byte*
0010 = 2	1101 = D	2D
0010 = 2	1101 = D	2D
0010 = 2	1101 = D	2D
0000 = 0	0101 = 5	05
0000 = 0	0000 = 0	00

The hexadecimal value for the first byte of vector information is 2D. The hexadecimal value of the second and third bytes of vector information is 2D also. The hexadecimal value of the fourth vector information is 05 and the last byte of vector information is 00, indicating that the definition of the shape is complete. The complete hexadecimal shape definition of a horizontal line that begins at the left end and extends seven dots to the right is: 2D 2D 2D 05 00.

Summary

You have followed these steps to create your shape:

STEP 1: Draw shape

STEP 2: Plot shape using dots

Dot	1	2	3	4	5	6	7
	.	.	.	.	.	.	.

STEP 3: Determine start and direction from each plotted point

Vector	1	2	3	4	5	6	7
	•→	•→	•→	•→	•→	•→	•→

STEP 4: Translate each plotted vector into binary code

Code	1	2	3	4	5	6	7
	101	101	101	101	101	101	101

STEP 5: Position binary codes into individual eight-bit bytes
(Vector 1 goes into column A, vector 2 in column B, etc.)

C	*B*	*A*		
	2	1		
00	101	101	=	00101101
	4	3		
00	101	101	=	00101101
	6	5		
00	101	101	=	00101101
		7		
00	000	101	=	00000101
00	000	000	=	00000000

STEP 6: Divide the eight bits in each byte into two parts

Left 4 Bits	*Right 4 Bits*
0010	1101
0010	1101
0010	1101
0000	0101
0000	0000

STEP 7: Translate each of the 4 bits into hexadecimal values

2D
2D
2D
05
00

SHAPE TABLE

You now have a hexadecimal definition of a shape (a short horizontal line). This definition must be placed in the computer's memory and the computer must be told where to put the definition in its memory. The shape definition can be located anywhere in free memory except the area of memory reserved for the high-resolution page you will be using. Different individuals use different areas for their shape-table information. I prefer the area of memory just above the second high-resolution page. That area of memory begins at location $6000 (or 24576 decimal).

You must also choose how you will enter the shape-table information. Applesoft BASIC cannot use hexadecimal values directly, so those values must be converted to decimal values.

Hex		*Decimal*
2D	=	45
2D	=	45
2D	=	45
05	=	5
00	=	00

After being converted to decimal values, the shape definition can be placed in memory using the POKE command. I will use this method most of the time, but it is not the only way to place shape-table information into the computer's memory.

The hexadecimal values can be directly placed into the computer's memory by telling the computer to enter the information directly from the monitor. Typing

```
CALL -151 {RETURN}
```

at the BASIC prompt places the computer into the monitor and provides you with an asterisk prompt (*). At this point, you can enter a memory location (using hexadecimal values), a colon and the hexadecimal values of the shape definition:

```
6000:01 00 04 00 2D 2D 2D 05 00 {RETURN}
```

Please notice that I have included four additional bytes of information at the beginning of the shape definition: 01 00 04 00. These four bytes contain information about the shape table itself. The first byte, 01, indicates that there is only one shape in this shape table. The second byte is unused (some commercial shape table programs do make use of this byte). The third byte (04) contains the location of the beginning of the shape-table definition relative to the beginning of the shape table itself. This value is called the index to the shape definition. The fourth byte (00) in this instance is also unused.

These nine bytes make up a shape table located at $6000 and containing the definition of a short horizontal line. In order to use this shape table, you must provide the computer with the location of the table. Memory locations E8 and E9 (232 and 233 decimal) are used to store the address of the shape table. E8 (232) must contain the low-order portion of the address (00 hex or 00 decimal) and E9 (233) must hold the high-order portion of the address (60 hex or 96 decimal) (E8:00 or POKE 232,0; E9:60 or POKE 233,96). If you have entered this information using the monitor, you can return to BASIC by typing:

```
3DOG {RETURN}
```

or if you are using a IIGS by typing the letter:

```
Q {RETURN}
```

To place all the shape-table information into the computer using BASIC instead of the monitor, enter and run the following short program. Nothing will appear on the screen but

the shape-table information will be included in memory. Save the program to diskette as HORIZ.LNE.SHAPE.

```
100 REM **--HORIZ.LNE.SHAPE--**
110:
120:
130 REM **--SHAPE TABLE ADDRESS--**
140 POKE 232,0
150 POKE 233,96
160:
170 REM **--SHAPE TABLE INFORMATION--**
180 FOR L = 24576 TO 24584:REM $6000
190 READ V
200 POKE L,V
210 NEXT L
220:
230:
500 REM **--END--**
510 END
520:
530:
700 REM **--SHAPE TABLE VALUES--**
710 DATA 1,0,4,0,45,45,45,5,0:REM HORIZ.LINE
```

After you run the program, the shape table will be located in the computer's memory and the shape-table address will be correctly set.

Once the shape-table information has been placed into the computer and the address of the shape table has been provided to locations 232 and 233, you can display the shape on either high-resolution screen.

Displaying the Shape

Enter the following BASIC program to display the horizontal-line shape and demonstrate some of the capabilities associated with shape tables and shape manipulation:

```
100 REM ***--HORIZ.LNE.DEMO--***
110:
120:
130 REM **--HIRES INITIALIZATION--**
140 HGR
150 HCOLOR = 3
160:
170:
180 REM **--SHAPE INITIALIZATION--**
190 SCALE = 1:REM ORIGINAL SIZE
```

```
200 ROT = 0:REM ORIGINAL ANGLE
210:
220:
500 REM **--DRAW SHAPE--**
510 DRAW 1 AT 100,50
520 GOSUB 900:REM PAUSE
525:
527 REM *--INCREASE SIZE--*
530 SCALE = 5
540 DRAW 1 AT 100,75
550 GOSUB 900:REM PAUSE
555:
557 REM *--ROTATE SHAPE--*
560 ROT = 16:REM ROTATE 90 DEGREES
570 DRAW 1 AT 100,50
580 GOSUB 900:REM PAUSE
585:
587 REM *--DRAW BORDERS W/SHAPE--*
590 ROT = 0:SCALE = 50
600 DRAW 1 AT 0,0
610 DRAW 1 AT 0,159
620 ROT = 16
630 DRAW 1 AT 0,0
640 DRAW 1 AT 279,0
650 GOSUB 900:REM PAUSE
655:
657 REM *--CREATE PATTERN--*
660 SCALE = 5
670 FOR I = 0 TO 64
680 ROT = I
690 DRAW 1 AT 200,50
700 NEXT I
710 GOSUB 900:REM PAUSE
715:
717 REM *--ERASE PATTERN--*
720 FOR I = 64 TO 0 STEP -1
730 ROT = I
740 DRAW 1 AT 200,50
750 XDRAW 1 AT 200,50
760 NEXT I
770 GOSUB 900:REM PAUSE
780:
790:
800 REM **--END--**
810 TEXT
820 END
830:
840:
```

```
900 REM **--PAUSE--**
910 INPUT "PRESS THE RETURN KEY TO CONTINUE:";L$
920 RETURN
```

If you read through the program listing, you should be able to see some of the possible ways you can work with shapes. With a very small horizontal-line shape and the Applesoft BASIC shape commands you can create a variety of displays (Figs. 8.3 through 8.6). SCALE sets the size of the shape (255 is maximum), and ROT (for rotate) establishes the angle at which the shape is to be drawn (0 = original angle, 16 = 90 degree turn, 32 = 180 degree turn, 64 = full rotation; each point = 5.625 degrees or one sixty-fourth of a circle). DRAW places the specified shape on the screen at a specified location, and XDRAW draws the specified shape at a specified location in the color that is the complement of the color currently at that location. XDRAW is often used, as in HORIZ.LNE.DEMO, to erase a previously drawn object.

Saving the Shape

Before proceeding with more information concerning the capabilities of shapes and shape manipulation, you should save the shape you created. The BLOAD and BSAVE commands can be used in the same way as shown in the section on storing information from the

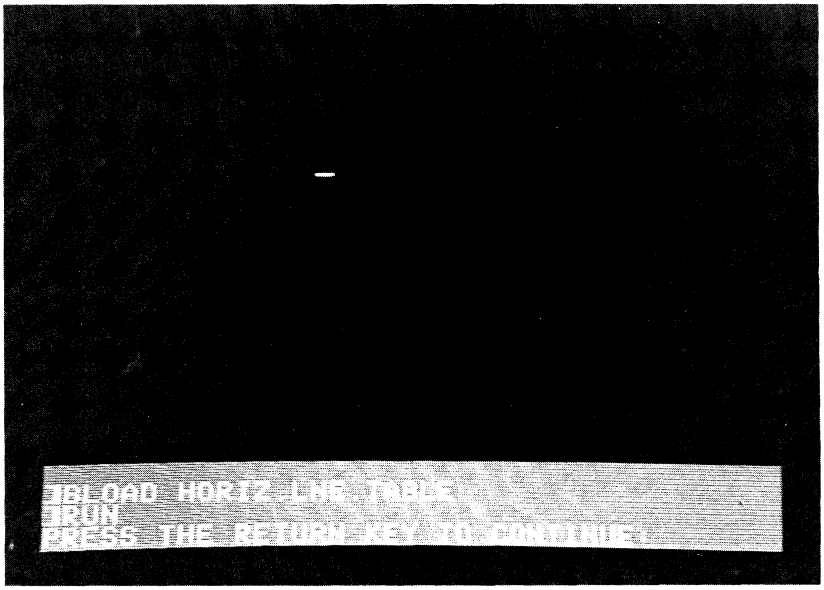


FIGURE 8.3. Horizontal line shape.

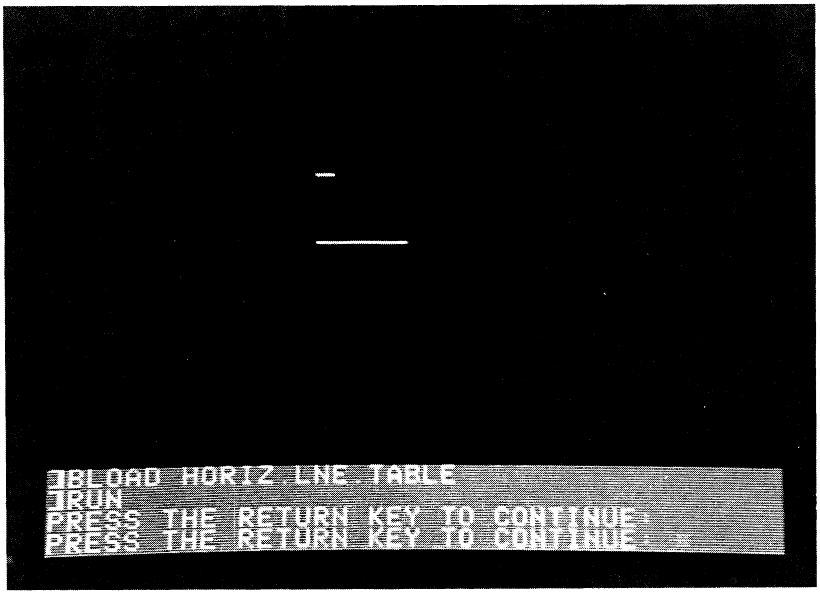


FIGURE 8.4. Horizontal line shape scaled to 5.

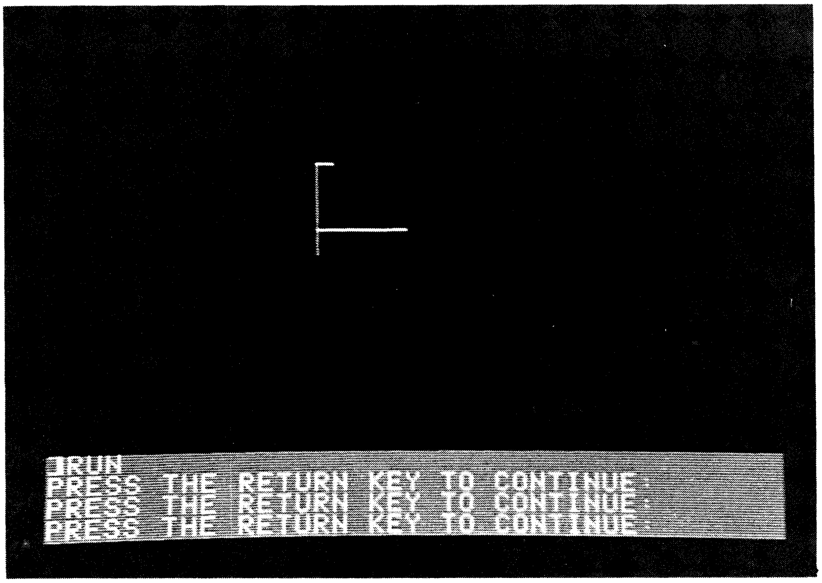


FIGURE 8.5. Horizontal line shape scaled to 5 and rotated to 16 (90 degrees).

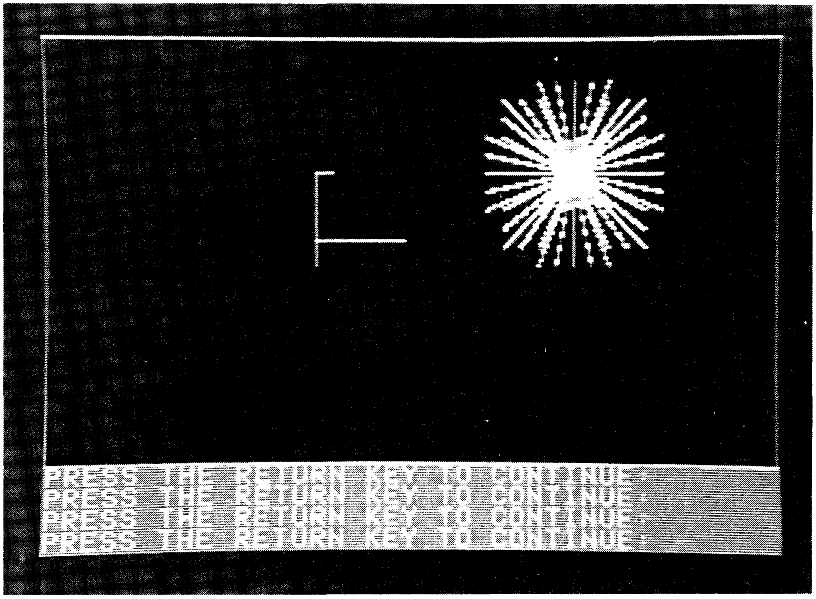


FIGURE 8.6. Horizontal line shape used to create borders and pattern.

high-resolution pages in Chapter 7. The shape information will be stored in a different location, so the parameters in the commands will be different. To save the horizontal-line shape, type

```
BSAVE HORIZ.LNE.TABLE, A$6000,L$9 {RETURN}
```

This command assumes you have somehow (either through the monitor or with the BASIC program) entered the shape information into a shape table located at memory location $6000. To retrieve a shape table enter the command

```
BLOAD HORIZ.LNE.TABLE {RETURN}
```

If you want to retrieve the shape table and display it in a different place than the one from which it was originally saved, you must include the new address in the BLOAD command:

```
BLOAD HORIZ.LNE.TABLE,A$6200
```

Either of these commands can be included in a BASIC program as long as they are preceded by a CHR$(4) (the ASCII value for Control D):

```
150 PRINT CHR$(4); "BSAVE HORIZ.LNE.TABLE,A$6000,L$9"
```

or

```
160 PRINT CHR$(4); "BLOAD HORIZ.LNE.TABLE"
```

MULTIPLE SHAPES IN A SINGLE TABLE

A shape table can contain information about more than one shape. Remember that the first four bytes in our example shape-table definition contained information about the shape table itself. The first byte in the shape table indicated the number of shapes in the table, and the third byte provided the starting location for the actual shape definition. By changing the contents of these two bytes you can include additional shape definitions in the table. In fact, a single shape table can contain up to 255 shape definitions.

In our shape table example, the fourth byte was a zero, since we had only one shape and that definition began immediately. If we have many definitions, two bytes may be needed to indicate the exact starting location of some of those definitions. In that case, each definition would have to be allowed two bytes for the index to its starting location. If the start of a shape definition is 256 bytes from the start of the table, the index for that definition would have a value of 00 in the first index byte and a 01 in the second index byte. If the start of a shape definition is 512 bytes from the start of the table, the index for that definition would have a value of 00 in the first index byte and a 02 in the second index byte. If the start of a shape definition is 517 bytes from the start of the table, the index for that definition would have a value of 05 in the first index byte and a 02 in the second index byte ($517 \div 256 = 2$ with a remainder of 5).

Any shape table actually contains three parts:

1. The number of shapes in the table
2. The address or index for each shape
3. The shape definitions

It is often a good idea to leave some space between the end of the second section and the beginning of the third section so that additional shapes can be added. To add an additional shape to the horizontal-line shape table requires repositioning the horizontal-line shape to include room for the address or index of the new shape or shapes. If you used the BASIC program given above to enter the shape table information, it will be relatively easy to alter it. Load the program back into the computer and make the following changes and additions:

```
100 REM **--MULTIPLE.SHAPES--**
180 FOR L = 24576 TO 24629:REM $6000
700 REM **--SHAPE TABLE VALUES--**
710 DATA 3,0:REM # OF SHAPES
720 DATA 20,0:REM START OF #1
730 DATA 25,0:REM START OF #2
740 DATA 34,0:REM START OF #3
745 DATA 0,0,0,0,0,0,0,0,0,0,0,0:REM ROOM FOR MORE SHAPES
```

```
750 DATA 45,45,45,5,0:REM HORIZ.LINE
760 DATA 45,45,54,54,63,63,36,36,0:REM SQUARE
770 REM *--NEXT SHAPE IS AIRPLANE--*
780 DATA 45,36,44,54,46,45,62,63,54,62
790 DATA 36,60,63,63,38,39,37,46,45,0
```

When you have finished making the changes and additions, save the new program as MULTIPLE.SHAPES and then run it. Again, you will not see anything on the screen. Remember to save the shape table by typing

```
BSAVE MULTI.SHP.TABLE,A$6000,L$36 {RETURN}
```

Next, load in the HORIZ.LNE.DEMO program and make the following changes:

```
100 REM ***--MULT.SHP.DEMO--***
540 DRAW 2 AT 100,75
560 ROT = 32:REM ROTATE 180 DEGREES
570 DRAW 3 AT 111,85
690 DRAW 2 at 200,50
740 DRAW 2 AT 200,50
750 XDRAW 2 AT 200,50
```

and add one additional instruction:

```
558 SCALE = 1
```

Save the program as MULTI.SHP.DEMO. When you run MULTI.SHP.DEMO, you should see the initial horizontal-line shape, the new square shape, the airplane shape inside the square, and the square used to create and erase a pattern. As you can see, more than one shape can easily be included in a shape table and then used from within a program (Figs. 8.7 through 8.9).

Multiple Shape Tables in Memory

Although the Apple can use shape definitions from only one table at a time, more than one shape table can be stored in memory at one time. Each shape table must contain all three sections: (1) the number of shapes in the table, (2) the address or index for each shape, and (3) the shape definitions. Before a shape from any table can be used, you must use POKE to put the address (exact memory location) for that table into locations 232 and 233. The use of multiple tables can be helpful in certain applications as long as the programmer keeps everything straight. Strange results will occur if you use the wrong shape from the wrong table.

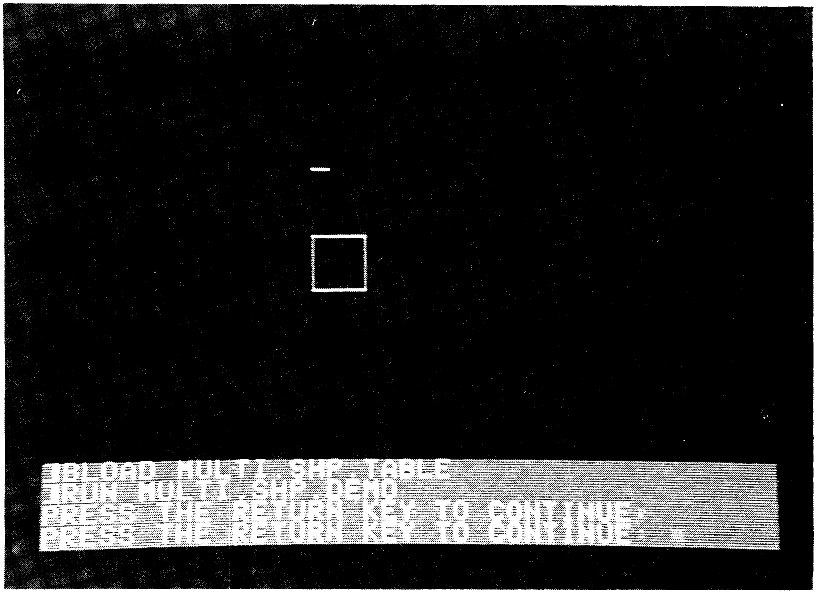


FIGURE 8.7. Multiple shapes in a single table—horizontal line shape and square shape.

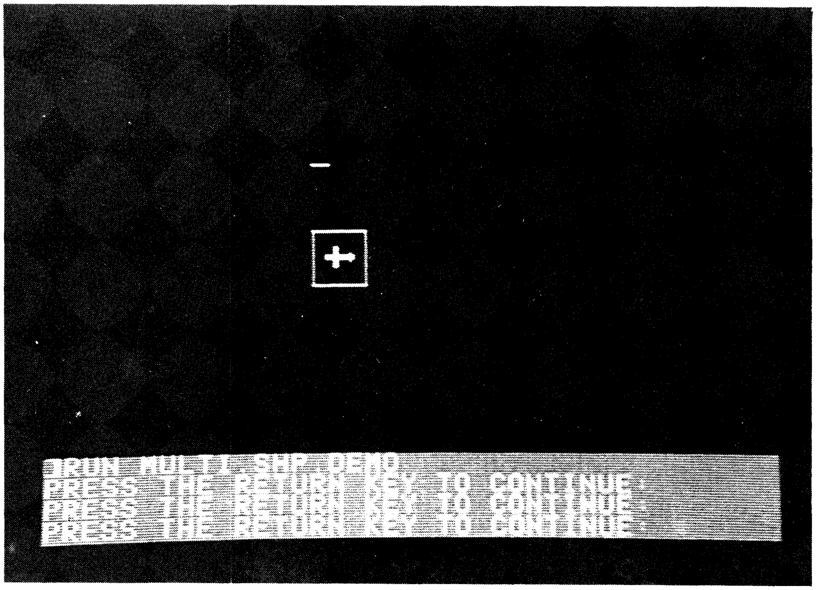


FIGURE 8.8. Multiple shapes in a single table—horizontal line shape, square shape, and airplane shape.

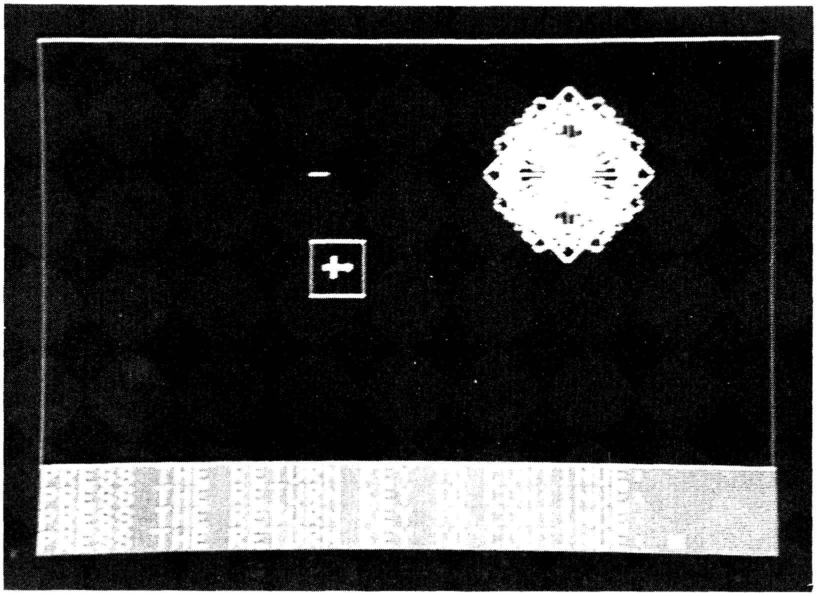


FIGURE 8.9. Multiple shapes in a single table – horizontal line shape, square shape, airplane shape, and pattern created with square shape.

High-Resolution Text

One of the most common uses of shape tables on the high-resolution screens actually involves text. Since text cannot be directly entered except on the text page, programmers use the shape-table method to create the shape of each character in the ASCII code. Once the shape table has been created, those characters can be displayed with the DRAW and XDRAW commands. I have included an ASCII-code shape table at the end of this chapter in two different formats. The first format provides the decimal values for each shape and the shape-table information and uses POKE to put these values into memory with a BASIC program. The second format provides hexadecimal values for the entire table. The hexadecimal values must be entered directly into the computer's memory through the monitor.

Whichever format you choose, it will take a considerable amount of typing to enter the information necessary to create an ASCII shape table. *Regardless of the way you enter the information, make sure that you save the table.* If you use the BASIC program, save it before running it. Save the table with the following command:

```
BSAVE ASCII.TEXT,A$6000,L$5FF
```

I have also included a program that will display a list of all the ASCII characters in the table on High-Resolution Page 1 (Fig. 8.10).

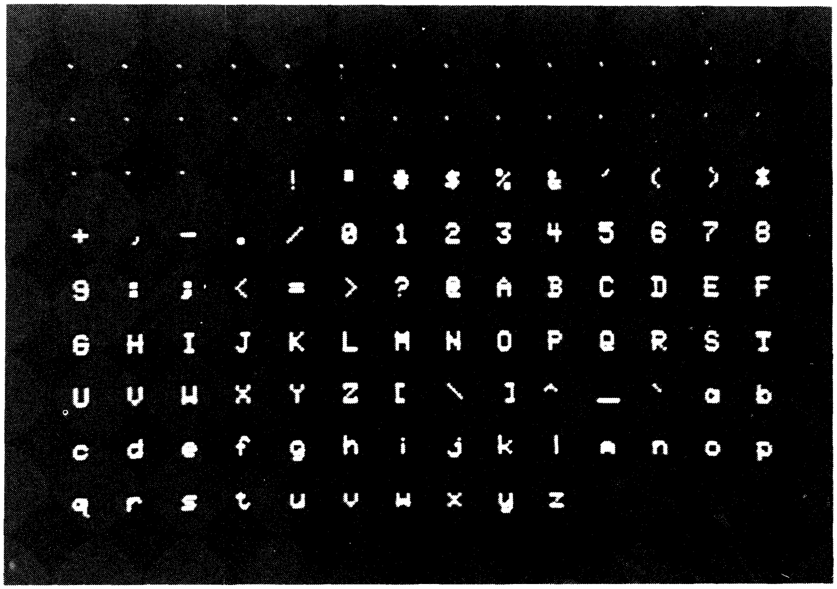


FIGURE 8.10. ASCII character shapes.

ANIMATION AND SHAPES

Animation with shapes is very common in arcade-type games. The techniques involved are the same as those used in Chapter 7. The shape is drawn in one location on the screen, then erased from that position and redrawn in a new location. The process continues until the desired movement has been completed. If flicker-free animation is desired, the page-flipping technique explained in Chapter 7 can also be used. Additional programs included at the end of this chapter demonstrate shape animation with and without page flipping. I have also included the definitions for several different shapes in addition to the definitions for the high-resolution character set (Fig. 8.11).

SHAPE.MAKER Program

The SHAPE.MAKER program can easily be used to create additional shapes using the up, down, left and right arrow keys to indicate vectors that plot and then move in the direction indicated by the arrow. The "I", "J", "K", and "L" keys are used to indicate vectors that do not plot but move either up ("I" key), down ("M" key), left ("J" key) or right ("K" key). All you need to do to create shapes is to draw out the shape using plotting and nonplotting

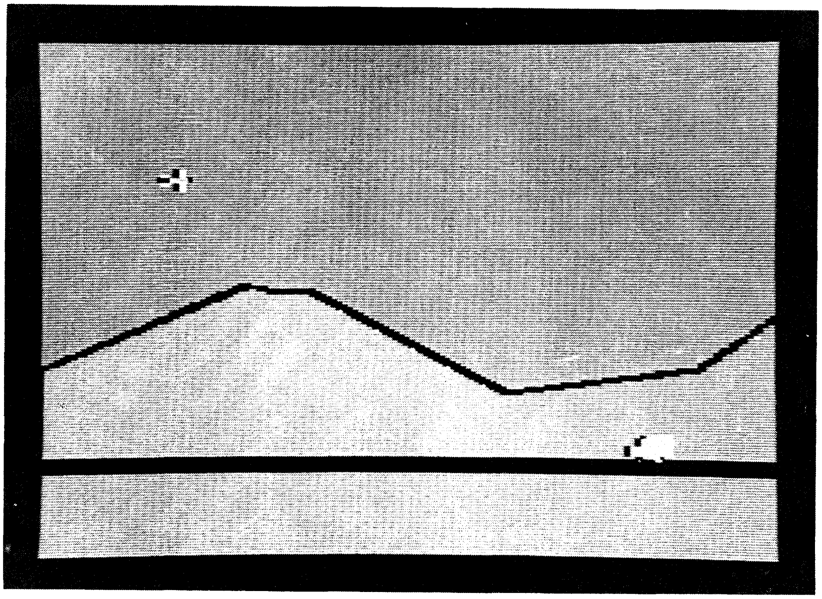


FIGURE 8.11. Animated airplane and truck shapes on multiple-colored background.

vectors. Then enter those vectors using the arrow keys and the letter keys. The rest of the work involved in shape creation is done by the computer although you must still save the resulting table or add the hex numbers to an existing table.

The SHAPE.MAKER program is easy to understand and use. It does not take long to type in—I left out certain features to keep it relatively short—yet it really reduces the work involved in shape creation. The SHAPE.MAKER program does not manipulate shape tables. If you wish to add, delete, or edit a shape you must either purchase a commercial program, create a shape-table manipulation program yourself or enter the information through a BASIC program or the monitor. The SHAPE.MAKER program is designed around the steps listed in this chapter and in the Apple manuals. As such, it is an instructional tool, and not the most efficient method of shape creation. Used carefully, however, it should significantly simplify the shape-creation process.

Collision Counter

Apple includes a memory location (234) that monitors the area affected by a shape. When two or more shapes overlap, the value in memory location 234 changes. By constantly testing location 234, you can determine if two or more shapes have "collided". When a collision occurs, the computer can be instructed to transfer control to a routine that displays the results of the collision.

Using the collision counter appears to be relatively easy, but it can produce some interesting results. For example, location 234 contains different values depending upon whether the shape was created with the DRAW command or the XDRAW command. In addition, if a background is used and shapes are created with XDRAW, location 234 will contain different values for some of the different backgrounds. Therefore, to use the collision counter you must know what value memory location 234 contains under a variety of circumstances. (You can examine the contents of memory location with the PEEK command— X = PEEK (234). If you want to see the value of X, then print X.) Once you have examined the values, you can determine when the collision occurs. To do this, you must use IF-THEN statements that test for the value that occurs when the collision takes place.

In the collision program, I use a third shape to simulate an explosion. Using XDRAW, the explosion shape is first drawn at the point of the collision in its original size. The explosion shape then is scaled up to a value of two and redrawn with the XDRAW command to give the impression of an expanding explosion. Additional effects, such as sound or shapes that represent images of destroyed or damaged objects, can be included in the collision routine (Figs. 8.12 and 8.13).

The explosion shape was added to the shape table containing the truck and plane, but could just as easily have been loaded into memory at a different location. You can use BLOAD to put both tables into memory at the start of the program. Memory locations 232 and 233 can quickly and easily be set (with the POKE command) to different locations for the different addresses of the two tables. In time-critical situations, keeping shapes in a

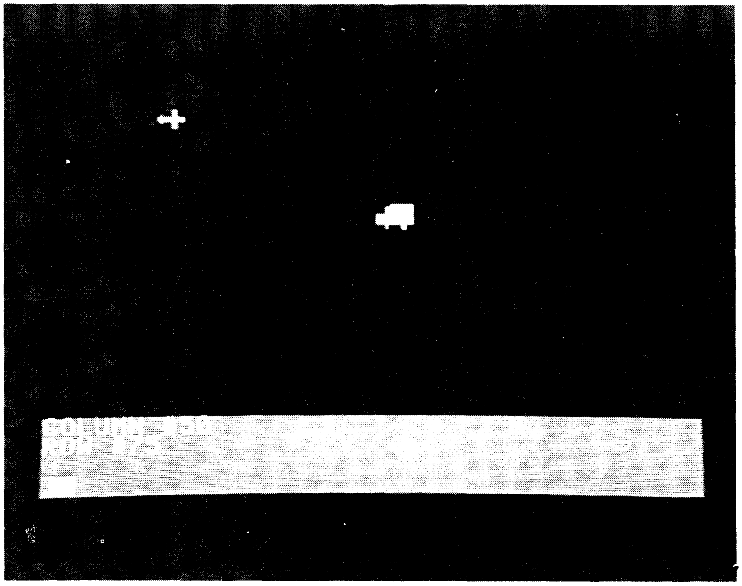


FIGURE 8.12. Animated airplane and truck shapes.

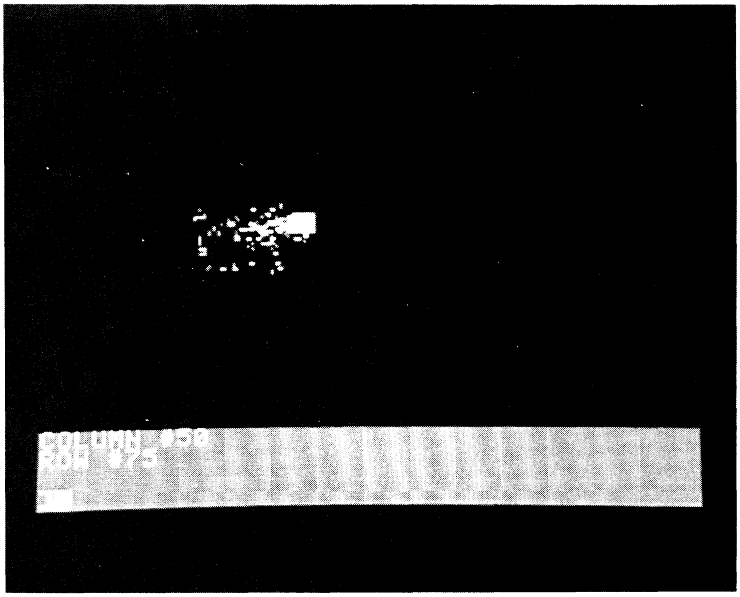


FIGURE 8.13. Animated airplane, truck, and explosion shapes.

single table will obviously involve fewer instructions and will make it possible to draw them faster, but for many tasks, switching between tables is an acceptable method of using multiple shapes.

Elaborate Shapes

I have included additional examples of shapes because the manner of drawing these shapes is both more complex than it may seem and more important than some authors suggest. Diagonal lines cannot be drawn directly and therefore must be created with a combination of horizontal and vertical vectors. Some shapes are best created from the center out, while other shapes must (by the nature of their use) be created by beginning on an outer edge. Some commercial shape-creation programs scan the screen after the user has indicated that the shape is finished. Shapes defined in this manner tend to exhibit horizontal or vertical scan lines when they are increased in size through use of the SCALE command. Animated shapes (a waving arm, for example) must be created as several separate shapes and then drawn in proper sequence.

Once again, I have covered a significant amount of information in this chapter. As I said at the beginning of Chapter 7, much can be written about high-resolution graphics, so it is very frustrating to have to limit the number of topics and examples included in these high-resolution chapters. My hope is that enough information and examples are included in an understandable way that you will be able to figure out aspects of graphics that are not

directly or completely explained or will be interested enough to go on to a more detailed reference book. In the next chapter, I will discuss the ways that high-resolution graphics can be used in different applications.

REVIEW QUESTIONS

1. What Applesoft BASIC words are used to position shapes on the high-resolution screen?
2. What Applesoft BASIC word is used to size a high-resolution shape?
3. What Applesoft BASIC word is used to turn a high-resolution shape around?
4. What term is used to describe where the shape begins and exactly how to construct the shape from that beginning point by indicating plot and movement instructions?
5. Once a shape has been defined, it must be placed in the computer's memory in the form of ______________.
6. What does the first piece of information in a shape table indicate?
7. Every shape definition must end with the last byte equal to ______________.
8. What two locations in memory are used to store the address of the shape table?
9. What command is used to save a shape table?
10. What command is used to load a shape table from diskette?

Horizontal-Line Shape

```
100  REM  ***--HORIZ.LNE.SHAPE--***
110 :
120 :
130  REM  **--SHAPE TABLE ADDRESS--**
140  POKE 232,0
150  POKE 233,96
160 :
170  REM  **--SHAPE TABLE INFORMATION--**
180  FOR L = 24576 TO 24584: REM  $6000
190  READ V
200  POKE L,V
210  NEXT L
220 :
230 :
500  REM   **--END--**
510  END
520 :
530 :
700  REM  **--SHAPE TABLE VALUES--**
710  DATA   1,0,4,0,45,45,45,5,0: REM  HORIZ.LINE
```

Horizontal-Line Demo

```
100  REM   ***--HORIZ.LNE.DEMO--***
110 :
120 :
130  REM  **--HIRES INITIALIZATION--**
140  HGR
150  HCOLOR= 3
160 :
170 :
180  REM  **--SHAPE INITIALIZATION--**
190  SCALE= 1: REM   ORIGINAL SIZE
200  ROT= 0: REM  ORIGINAL ANGLE
210 :
220 :
230 :
500  REM  **--DRAW SHAPE--**
510  DRAW 1 AT 100,50
520  GOSUB 900: REM  PAUSE
525 :
527  REM   *--INCREASE SIZE--*
530  SCALE= 5
540  DRAW 1 AT 100,75
```

```
550  GOSUB 900: REM  PAUSE
555 :
557  REM  *--ROTATE SHAPE--*
560  ROT= 16: REM  ROTATE 90 DEGREES
570  DRAW 1 AT 100,50
580  GOSUB 900: REM   PAUSE
585 :
587  REM   *--DRAW BORDERS W/SHAPE--*
590  ROT= 0: SCALE= 50
600  DRAW 1 AT 0,0
610  DRAW 1 AT 0,159
620  ROT= 16
630  DRAW 1 AT 0,0
640  DRAW 1 AT 279,0
650  GOSUB 900: REM  PAUSE
655 :
657  REM  *--CREATE PATTERN--*
660  SCALE= 5
670  FOR I = 0 TO 64
680  ROT= I
690  DRAW 1 AT 200,50
700  NEXT I
710  GOSUB 900: REM  PAUSE
715 :
717  REM  *--ERASE PATTERN--*
720  FOR I = 64 TO 0 STEP  - 1
730  ROT= 1
740  DRAW 1 AT 200,50
750  XDRAW 1 AT 200,50
760  NEXT I
770  GOSUB 900: REM  PAUSE
780 :
790 :
800  REM  **--END--**
810  TEXT
820  END
830 :
840 :
900  REM  **--PAUSE--**
910  INPUT "PRESS THE RETURN KEY TO CONTINUE: ";L$
920  RETURN
```

Multiple-Shape Table

```
100  REM   ***--MULTIPLE.SHAPES--***
110 :
```

```
120 :
130  REM  **--SHAPE TABLE ADDRESS--**
140  POKE 232,0
150  POKE 233,96
160 :
170  REM  **--SHAPE TABLE INFORMATION--**
180  FOR L = 24576 TO 24629: REM   $6000
190  READ V
200  POKE L,V
210  NEXT L
220 :
230 :
500  REM   **--END--**
510  END
520 :
530 :
700  REM  **--SHAPE TABLE VALUES--**
710  DATA  3,0: REM  # OF SHAPES
720  DATA  20,0: REM  START OF #1
730  DATA  25,0: REM  START OF #2
740  DATA  34,0: REM  START OF #3
745  DATA  0,0,0,0,0,0,0,0,0,0,0,0: REM  ROOM FOR MORE SHAPES
750  DATA  45,45,45,5,0: REM  HORIZ.LINE
760  DATA  45,45,54,54,63,63,36,36,0: REM  SQUARE
770  REM  *--NEXT SHAPE IS AIRPLANE--*
780  DATA  45,36,44,54,46,45,62,63,54,62
790  DATA  36,60,63,63,38,39,37,46,45,0
```

Multiple-Shape Demo

```
100  REM    ***--MULTI.SHP.DEMO--***
110 :
120 :
130  REM  **--HIRES INITIALIZATION--**
140  HGR
150  HCOLOR= 3
160 :
170 :
180  REM  **--SHAPE INITIALIZATION--**
190  SCALE= 1: REM   ORIGINAL SIZE
200  ROT= 0: REM  ORIGINAL ANGLE
210 :
220 :
230 :
500  REM  **--DRAW SHAPE--**
510  DRAW 1 AT 100,50
```

```
520  GOSUB 900: REM  PAUSE
525 :
527  REM   *--INCREASE SIZE--*
530  SCALE= 5
540  DRAW 2 AT 100,75
550  GOSUB 900: REM  PAUSE
555 :
557  REM  *--ROTATE SHAPE--*
558  SCALE= 1
560  ROT= 32: REM  ROTATE 180 DEGREES
570  DRAW 3 AT 111,85
580  GOSUB 900: REM  PAUSE
585 :
587  REM   *--DRAW BORDERS W/SHAPE--*
590  ROT= 0: SCALE= 50
600  DRAW 1 AT 0,0
610  DRAW 1 AT 0,159
620  ROT= 16
630  DRAW 1 AT 0,0
640  DRAW 1 AT 279,0
650  GOSUB 900: REM  PAUSE
655 :
657  REM  *--CREATE PATTERN--*
660  SCALE= 5
670  FOR I = 0 TO 64
680  ROT= I
690  DRAW 2 AT 200,50
700  NEXT I
710  GOSUB 900: REM  PAUSE
715 :
717  REM  *--ERASE PATTERN--*
720  FOR I = 64 TO 0 STEP  - 1
730  ROT= I
740  DRAW 2 AT 200,50
750  XDRAW 2 AT 200,50
760  NEXT I
770  GOSUB 900: REM  PAUSE
780 :
790 :
800  REM  **--END--**
810  TEXT
820  END
830 :
840 :
900  REM  **--PAUSE--**
910  INPUT "PRESS THE RETURN KEY TO CONTINUE: ";L$
920  RETURN
```

ASCII Shape Table

```
100  REM   ***--ASCII.SHAPES--***
110 :
120 :
130  REM  **--SHAPE TABLE ADDRESS--**
140  DATA  0,96
150 :
160  REM  **--NUMBER OF SHAPES IN TABLE--**
170  DATA  150,0
180 :
190  REM  **--SHAPE 1 INDEX--**
200  DATA  0,0
210 :
220  REM  **--SHAPE 2 INDEX--**
240  DATA  0,0
250 :
260  REM  **--SHAPE 3 INDEX--**
270  DATA  0,0
280 :
290  REM  **--SHAPE 4 INDEX--**
300  DATA  0,0
310 :
320  REM  **--SHAPE 5 INDEX--**
330  DATA  0,0
340 :
350  REM **--SHAPES 6 TO 32 INDEX--**
360  DATA  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
370 :
372  REM  **--INDEX FOR SPACE--**
374  DATA 229,5
376 :
380  REM  **--INDEX FOR ! --**
390  DATA 46,1
400 :
410  REM  **--INDEX FOR " --**
420  DATA 51,1
430 :
440  REM  **--INDEX FOR # --**
450  DATA 60,1
460 :
470  REM  **--INDEX FOR $ --**
480  DATA 79,1
490 :
500  REM  **--INDEX FOR % --**
510  DATA 94,1
520 :
```

```
530  REM  **--INDEX FOR & --**
540  DATA 107,1
550 :
560  REM  **--INDEX FOR ' --**
570  DATA 126,1
580 :
590  REM  **--INDEX FOR ( --**
600  DATA 145,1
610 :
620  REM  **--INDEX FOR ) --**
630  DATA 153,1
640 :
650  REM  **--INDEX FOR * --**
660  DATA 161,1
670 :
680  REM  **--INDEX FOR + --**
690  DATA 183,1
700 :
710  REM  **--INDEX FOR , --**
720  DATA 192,1
730 :
740  REM  **--INDEX FOR - --**
750  DATA 198,1
760 :
770  REM  **--INDEX FOR . --**
780  DATA 203,1
790 :
800  REM  **--INDEX FOR / --**
810  DATA 222,1
820 :
830  REM  **--INDEX FOR 0 --**
840  DATA 231,1
850 :
860  REM  **--INDEX FOR 1 --**
870  DATA 249,1
880 :
890  REM  **--INDEX FOR 2 --**
900  DATA 13,2
910 :
920  REM  **--INDEX FOR 3 --**
930  DATA 33,2
940 :
950  REM  **--INDEX FOR 4 --**
960  DATA 43,2
970 :
980  REM  **--INDEX FOR 5 --**
990  DATA 53,2
1000 :
```

```
1010  REM  **--INDEX FOR 6 --**
1020  DATA 74,2
1030 :
1040  REM  **--INDEX FOR 7 --**
1050  DATA 94,2
1060 :
1070  REM  **--INDEX FOR 8 --**
1080  DATA 102,2
1090 :
1100  REM  **--INDEX FOR 9 --**
1110  DATA 115,2
1120 :
1130  REM  **--INDEX FOR : --**
1140  DATA 135,2
1150 :
1160  REM  **--INDEX FOR ; --**
1170  DATA 154,2
1180 :
1190  REM  **--INDEX FOR < --**
1200  DATA 174,2
1210 :
1220  REM  **--INDEX FOR = --**
1230  DATA 183,2
1240 :
1250  REM  **--INDEX FOR > --**
1260  DATA 191,2
1270 :
1280  REM  **--INDEX FOR ? --**
1290  DATA 199,2
1300 :
1310  REM  **--INDEX FOR @ --**
1320  DATA 218,2
1330 :
1340  REM  **--INDEX FOR A --**
1350  DATA 232,2
1360 :
1370  REM  **--INDEX FOR B --**
1380  DATA 247,2
1390 :
1400  REM  **--INDEX FOR C --**
1410  DATA 11,3
1420 :
1430  REM  **--INDEX FOR D --**
1440  DATA 23,3
1450 :
1460  REM  **--INDEX FOR E --**
1470  DATA 43,3
1480 :
```

```
1490  REM  **--INDEX FOR F--**
1500  DATA 57,3
1510 :
1520  REM  **--INDEX FOR G--**
1530  DATA 69,3
1540 :
1550  REM  **--INDEX FOR H --**
1560  DATA 89,3
1570 :
1580  REM  **--INDEX FOR I --**
1590  DATA 102,3
1600 :
1610  REM  **--INDEX FOR J --**
1620  DATA 111,3
1630 :
1640  REM  **--INDEX FOR K --**
1650  DATA 131,3
1660 :
1670  REM  **--INDEX FOR L --**
1680  DATA 147,3
1690 :
1700  REM  **--INDEX FOR M --**
1710  DATA 154,3
1720 :
1730  REM  **--INDEX FOR N --**
1740  DATA 170,3
1750 :
1760  REM  **--INDEX FOR O --**
1770  DATA 185,3
1780 :
1790  REM  **--INDEX FOR P --**
1800  DATA 196,3
1810 :
1820  REM  **--INDEX FOR Q --**
1830  DATA 208,3
1840 :
1850  REM  **--INDEX FOR R --**
1860  DATA 223,3
1870 :
1880  REM  **--INDEX FOR S --**
1890  DATA 238,3
1900 :
1910  REM  **--INDEX FOR T --**
1920  DATA 251,3
1930 :
1940  REM  **--INDEX FOR U--**
1950  DATA 15,4
1960 :
```

```
1970  REM  **--INDEX FOR V --**
1980  DATA 24,4
1990 :
2000  REM  **--INDEX FOR W --**
2010  DATA 34,4
2020 :
2030  REM  **--INDEX FOR X --**
2040  DATA 55,4
2050 :
2060  REM  **--INDEX FOR Y --**
2070  DATA 67,4
2080 :
2090  REM  **--INDEX FOR Z --**
2100  DATA 76,4
2110 :
2120  REM  **--INDEX FOR LEFT BRACKET--**
2130  DATA 97,4
2140 :
2150  REM  **--INDEX FOR  BACK SLASH --**
2160  DATA 106,4
2170 :
2180  REM  **--INDEX FOR RIGHT BRACKET--**
2190  DATA 114,4
2200 :
2210  REM  **--INDEX FOR ^ --**
2220  DATA 123,4
2230 :
2240  REM  **--INDEX FOR UNDERLINE--**
2250  DATA 142,4
2260 :
2262  REM  **--INDEX FOR SHAPE #96--**
2264  DATA 151,4
2266 :
2270  REM  **--INDEX FOR SMALL A --**
2280  DATA 155,4
2290 :
2300  REM  **--INDEX FOR SMALL B --**
2310  DATA 168,4
2320 :
2330  REM  **--INDEX FOR SMALL C --**
2240  DATA 181,4
2350 :
2360  REM  **--INDEX FOR SMALL D --**
2370  DATA 194,4
2380 :
2390  REM  **--INDEX FOR SMALL E --**
2400  DATA 207,4
2410 :
```

```
2420  REM  **--INDEX FOR SMALL F --**
2430  DATA 219,4
2440 :
2450  REM  **--INDEX FOR SMALL G --**
2460  DATA 230,4
2470 :
2480  REM  **--INDEX FOR SMALL H --**
2490  DATA 246,4
2500 :
2510  REM  **--INDEX FOR SMALL I --**
2520  DATA 0,5
2530 :
2540  REM  **--INDEX FOR SMALL J --**
2550  DATA 4,5
2560 :
2570  REM  **--INDEX FOR SMALL K --**
2580  DATA 26,5
2590 :
2600  REM  **--INDEX FOR SMALL L --**
2610  DATA 39,5
2620 :
2630  REM  **--INDEX FOR SMALL M --**
2640  DATA 46,5
2650 :
2660  REM  **--INDEX FOR SMALL N --**
2670  DATA 60,5
2680 :
2690  REM  **--INDEX FOR SMALL O --**
2700  DATA 70,5
2710 :
2720  REM  **--INDEX FOR SMALL P --**
2730  DATA 80,5
2740 :
2750  REM  **--INDEX FOR SMALL Q --**
2760  DATA 94,5
2770 :
2780  REM  **--INDEX FOR SMALL R --**
2790  DATA 116,5
2800 :
2810  REM  **--INDEX FOR SMALL S --**
2820  DATA 125,5
2830 :
2840  REM  **--INDEX FOR SMALL T --**
2850  DATA 137,5
2860 :
2870  REM  **--INDEX FOR SMALL U --**
2880  DATA 158,5
2890 :
```

```
2900  REM  **--INDEX FOR SMALL V --**
2910  DATA 168,5
2920 :
2930  REM  **--INDEX FOR SMALL W --**
2940  DATA 179,5
2950 :
2960  REM  **--INDEX FOR SMALL X --**
2970  DATA 191,5
2980 :
2990  REM  **--INDEX FOR SMALL Y --**
3000  DATA 206,5
3010 :
3020  REM  **--INDEX FOR SMALL Z --**
3030  DATA 218,5
3040 :
4970 :
4980 :
4990 :
5000  REM  **--SHAPE DATA--**
5010 :
5020  REM  **-- ! --**
5030  DATA  49,54,182,77,0
5040 :
5050  REM  **-- " --**
5060  DATA  49,54,36,76,54,38,36,6,0
5070 :
5080  REM  **-- # --**
5090  DATA  49,54,54,38,44,63,45,45,55,38,36,36,52,62,63,45,45,7,0
5100 :
5110  REM  **-- $ --**
5120  DATA  9,54,54,54,60,47,45,12,28,63,28,12,45,61,0
5130 :
5140  REM  **-- % --**
5150  DATA  53,39,77,17,30,30,30,30,78,33,53,39,0
5160 :
5170  REM  **-- & --**
5180  DATA  105,9,26,27,31,110,77,26,27,59,106,13,21,27,31,115,109,21,0
5190 :
5200  REM  **-- ' --**
5210  DATA   73,13,26,27,31,10,77,17,27,27,83,73,17,27,27,83,73,17,0
5220 :
5230  REM  **-- ( --**
5240  DATA  9,30,30,54,14,14,28,0
5250 :
5260  REM  **-- ) --**
5270  DATA  9,14,14,54,30,30,12,0
5280 :
5290  REM  **-- * --**
```

```
5300  DATA  9,54,54,54,52,36,244,30,12,12,12,12,30,30,28,28,14,14,14,14,
    28,0
5310 :
5320  REM  **-- + --**
5330  DATA  137,54,54,36,63,45,45,7,0
5340 :
5350  REM  **-- , --**
5360  DATA  73,146,50,30,12,0
5370 :
5380  REM  **-- - --**
5390  DATA  146,45,45,7,0
5400 :
5410  REM  **-- . --**
5420  DATA  73,9,26,27,27,74,73,26,27,27,74,73,26,27,63,10,109,17,0
5430 :
5440  REM  **-- / --**
5450  DATA  73,17,30,30,30,30,30,12,0
5460 :
5470  REM  **-- 0 --**
5480  DATA  41,117,54,54,30,63,28,36,36,12,45,14,30,30,30,30,12,0
5490 :
5500  REM  **-- 1 --**
5510  DATA  9,77,26,27,63,74,77,26,27,31,74,77,26,27,31,10,45,13,2,0
5520 :
5530  REM  **-- 2 --**
5540  DATA  41,109,26,31,27,78,9,21,27,63,83,77,17,27,27,51,45,45,21,0
5550 :
5560  REM  **-- 3 --**
5570  DATA  45,45,246,30,117,246,63,28,14,0
5580 :
5590  REM  **-- 4 --**
5600  DATA  54,46,45,61,36,52,54,54,38,0
5610 :
5620  REM  **-- 5 --**
5630  DATA 45,45,21,27,27,51,45,109,26,31,27,74,9,21,59,27,115,45,13,2,0
5640 :
5650  REM  **-- 6 --**
5660  DATA 41,109,26,31,27,110,73,26,59,63,110,9,21,59,27,115,45,13,2,0
5670 :
5680  REM  **-- 7 --**
5690  DATA  45,45,246,30,30,54,4,0
5700 :
5710  REM  **-- 8 --**
5720  DATA  41,117,246,63,30,118,45,12,228,63,28,100,0
5730 :
5740  REM  **-- 9 --**
5750  DATA  41,109,26,31,27,110,9,21,59,63,87,73,21,59,27,115,45,13,2,0
5760 :
```

```
5770  REM   **-- : --**
5780  DATA   73,9,26,27,63,10,109,17,27,27,83,109,17,27,59,87,73,17,0
5790 :
5800  REM   **-- ; --**
5810  DATA   73,9,26,27,63,10,109,17,27,27,83,109,17,27,59,23,109,9,2,0
5820 :
5830  REM   **-- < --**
5840  DATA  73,30,30,30,14,14,14,28,0
5850 :
5860  REM   **-- = --**
5870  DATA  18,45,45,22,63,63,5,0
5880 :
5890  REM   **-- > --**
5900  DATA   113,14,14,30,30,30,12,0
5910 :
5920  REM   **-- ? --**
5930  DATA  41,109,26,31,27,78,9,21,27,63,83,77,17,27,27,83,77,17,0
5940 :
5950  REM   **-- @ --**
5960  DATA  41,117,54,30,39,52,222,36,52,54,118,45,61,0
5970 :
5980  REM   **-- A --**
5990  DATA  9,14,14,54,54,36,60,63,55,54,36,36,12,30,0
6000 :
6010  REM   **-- B --**
6020  DATA  45,109,26,31,59,10,77,21,27,63,87,77,21,59,27,23,45,109,2,0
6030 :
6040  REM   **-- C --**
6050  DATA  73,14,28,63,30,54,54,14,45,12,30,0
6060 :
6070  REM   **-- D --**
6080  DATA  45,109,26,31,59,10,77,21,59,27,87,77,21,59,27,23,45,109,2,0
6090 :
6100  REM   **-- E --**
6110  DATA  45,45,63,63,54,46,45,63,55,54,45,45,7,0
6120 :
6130  REM   **-- F --**
6140  DATA  45,45,63,63,54,46,45,63,55,54,4,0
6150 :
6160  REM   **-- G --**
6170  DATA  41,109,26,31,27,110,73,26,63,31,110,9,21,59,27,115,45,13,2,0
6180 :
6190  REM   **-- H --**
6200  DATA  54,54,54,36,44,45,37,36,54,54,54,4,0
6210 :
6220  REM   **-- I --**
6230  DATA  9,61,47,54,54,54,47,61,0
6240 :
```

```
6250  REM  **-- J --**
6260  DATA  9,45,21,27,31,83,9,13,26,59,27,74,105,26,59,27,14,109,17,0
6270 :
6280  REM  **-- K --**
6290  DATA  54,54,54,36,44,12,12,12,30,30,30,14,14,14,20,0
6300 :
6310  REM  **-- L --**
6320  DATA  54,54,54,45,45,7,0
6330 :
6340  REM  **-- M --**
6350  DATA  54,54,54,36,36,36,14,14,38,12,12,54,54,54,36,0
6360 :
6370  REM  **-- N --**
6380  DATA  54,54,54,36,36,36,118,14,14,14,38,36,36,52,0
6390 :
6400  REM  **-- O --**
6410  DATA  41,117,54,54,30,63,28,36,36,12,0
6420 :
6430  REM  **-- P --**
6440  DATA  54,54,54,36,36,36,45,117,246,63,55,0
6450 :
6460  REM  **-- Q --**
6470  DATA  41,117,54,222,14,14,28,12,22,31,231,36,36,76,0
6480 :
6490  REM  **-- R --**
6500  DATA  54,54,54,36,36,36,45,117,246,63,14,14,14,28,0
6510 :
6520  REM  **-- S --**
6530  DATA  41,117,28,63,30,118,45,14,246,63,28,14,0
6540 :
6550  REM  **-- T --**
6560  DATA  45,45,21,59,59,115,105,17,27,59,83,105,17,27,59,83,45,13,2,0
6570 :
6580  REM  **-- U --**
6590  DATA  54,54,118,45,12,36,36,52,0
6600 :
6610  REM  **-- V --**
6620  DATA  54,54,14,14,12,12,36,36,6,0
6630 :
6640  REM  **-- W --**
6650  DATA  77,41,26,31,27,110,9,21,59,27,51,13,13,21,59,31,55,77,41,2,0
6660 :
6670  REM  **-- X --**
6680  DATA  118,14,14,14,254,27,100,12,12,12,52,0
6690 :
6691  REM  **-- Y --**
```

```
6692  DATA  118,14,54,38,36,12,12,52,0
6693 :
6694  REM   **-- Z --**
6695  DATA 45,45,21,59,27,115,9,13,26,27,31,10,77,17,59,27,51,45,45,21,0
6696 :
6697  REM   **--LEFT BRACKET--**
6698  DATA 45,63,54,54,54,45,7,0
6699 :
6700  REM   **--BACK SLASH --**
6701  DATA  18,14,14,14,14,14,7,0
6702 :
6703  REM   **--RIGHT BRACKET--**
6704  DATA 73, 73,45,54,54,54,63,7,0
6705 :
6706  REM   **-- ^ --**
6707  DATA  9,77,26,59,59,106,9,21,27,27,83,73,17,27,27,83,73,17,0
6708 :
6709  REM   **--UNDERLINE--**
6710  DATA  146,146,146,47,45,45,45,7,0
6711 :
6712  REM   **--SHAPE #96--**
6713  DATA   18,14,14,7,0
6714 :
6715 :
6716 :
6720  REM   **--SMALL LETTERS--**
6730 :
6740  REM   **-- A --**
6750  DATA  146,12,117,14,36,54,54,36,62,30,231,52,0
6760 :
6770  REM   **-- B --**
6780  DATA  54,54,54,36,12,12,117,54,30,231,28,4,0
6790 :
6800  REM   **-- C --**
6810  DATA 146,12,45,14,28,63,30,54,14,45,12,30,0
6820 :
6830  REM   **-- D --**
6840  DATA  73,49,54,54,38,247,231,36,12,45,14,54,0
6850 :
6860  REM   **-- E --**
6870  DATA  146,34,12,45,14,62,63,55,14,45,7,0
6880 :
6890  REM   **-- F --**
6900  DATA  146,45,39,100,117,28,247,54,54,38,0
6910 :
6920  REM   **-- G --**
```

```
6930  DATA  146,12,45,14,54,38,247,231,36,54,14,77,246,63,5,0
6940 :
6950  REM  **-- H --**
6960  DATA  54,54,54,36,12,12,117,54,38,0
6970 :
6980  REM  **-- I --**
6990  DATA  137,22,54,38,0
7000 :
7010  REM  **-- J --**
7020  DATA 18,73,41,26,27,27,74,41,21,59,27,83,73,21,59,27,115,45,13,2,0
7030 :
7040  REM  **-- K --**
7050  DATA  54,54,54,36,45,12,12,30,30,14,14,28,28,0
7060 :
7070  REM  **-- L --**
7080  DATA 137,54,54,54,4,0
7090 :
7100  REM  **-- M --**
7110  DATA  146,54,38,36,12,14,54,36,38,12,14,54,38,0
7120 :
7130  REM  **-- N --**
7140  DATA  18,54,54,36,44,12,117,54,38,0
7150 :
7160  REM  **-- O --**
7170  DATA  146,12,45,14,54,30,63,28,36,6,0
7180 :
7190  REM  **-- P --**
7200  DATA 146,54,54,54,36,44,14,101,36,28,63,30,6,0
7210 :
7220  REM  **-- Q --**
7230  DATA 18,73,9,26,31,63,106,41,21,59,31,115,109,21,59,27,83,73,21,21
     ,0
7240 :
7250  REM  **-- R --**
7260  DATA  18,54,54,36,44,12,117,28,0
7270 :
7280  REM  **-- S --**
7290  DATA  146,12,45,61,63,30,14,45,14,30,63,47,0
7300 :
7310  REM  **-- T --**
7320  DATA 18,105,9,26,27,63,14,77,17,27,27,87,77,21,27,63,83,73,17,0
7330 :
7340  REM  **-- U --**
7350  DATA  18,54,118,101,12,36,54,54,4,0
7360 :
7370  REM  **-- V --**
7380  DATA  18,54,14,14,28,14,12,12,36,6,0
```

```
7390 :
7400  REM   **-- W --**
7410  DATA  18,54,54,12,12,14,14,36,36,38,6,0
7420 :
7430  REM   **-- X --**
7440  DATA  18,14,14,14,14,28,28,30,30,12,12,12,12,30,0
7450 :
7460  REM   **-- Y --**
7470  DATA  18,54,118,101,12,36,54,54,246,63,5,0
7480 :
7490  REM   **-- Z --**
7500  DATA  18,45,45,30,30,30,30,45,45,7,0
7510 :
7512  REM   **--SPACE--**
7514  DATA  146,146,146,0
7516 :
7520  REM **--END OF TABLE--**
7530  DATA  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0
7540 :
10000  REM   **--POKE VALUES--**
10010  READ L
10020  READ H
10030  FOR I = 24576 TO 24821
10040  READ A
10050  POKE I,A
10060  NEXT I
10070 :
10080  FOR I = 24878 TO 26100
10090  READ B
10100  POKE I,B
10110  NEXT I
10970 :
10980 :
10990 :
12000  REM   **--DISPLAY CHARACTERS--**
12005  POKE 232,0: POKE 233,96
12010  HGR2
12020  HCOLOR= 3
12030  SCALE= 1
12040  ROT= 0
12045 X = 2:Y = 5
12050  FOR I = 1 TO 124
12060  DRAW I AT X,Y
12070 X = X + 20
12080  IF X > 270 THEN X = 2:Y = Y + 20
12090  NEXT I
```

ASCII Shape Demo

```
100  REM   ***--ASCII.SHP.DEMO--**
110 :
120 :
130  REM  **--INITIALIZE HIRES SHAPES--**
140  POKE 232,0: POKE 233,96
150  HGR2
160  HCOLOR= 3
170  SCALE= 1
180  ROT= 0
190 X = 2:Y = 5
200 :
210 :
500  REM   **--DISPLAY CHARACTERS--**
510  FOR I = 1 TO 124
520  DRAW I AT X,Y
530 X = X + 20
540  IF X > 270 THEN X = 2:Y = Y + 20
550  NEXT I
560 :
570 :
700  REM  **--END--**
710  INPUT "";L$
720  TEXT
730  END
```

Shape Maker

```
100  REM   ***--SHAPE.MAKER--***
110 :
120 :
130  REM  **--INITIALIZATION--**
140  DIM L$(50),H$(50),W$(50),HX$(50)
150 I = 1
160 J = 1
170 :
180 :
190  REM  **--GET ARROW KEYS OR I,J,K,M KEYS--**
200 KEY =  PEEK ( - 16384)
210  IF (KEY - 128) = 11 THEN A$(I) = "100": GOTO 500
220  IF (KEY - 128) = 21 THEN A$(I) = "101": GOTO 500
230  IF (KEY - 128) = 10 THEN A$(I) = "110": GOTO 500
240  IF (KEY - 128) =  8 THEN A$(I) = "111": GOTO 500
250  IF (KEY - 128) = 73 THEN A$(I) = "000": GOTO 500
260  IF (KEY - 128) = 75 THEN A$(I) = "001": GOTO 500
```

```
270  IF (KEY - 128) = 77 THEN A$(I) = "010": GOTO 500
280  IF (KEY - 128) = 74 THEN A$(I) = "011": GOTO 500
290  IF (KEY - 128) = 27 THEN 1000
300  GOTO 200
310 :
320 :
500  REM  **--CHECK VALUES & ADD--**
510  POKE  - 16368,0: REM  RESET KEYBOARD STROB
520  IF I = 3 THEN 700
530  IF I = 2 AND A$(I) = "000" THEN 700
540 L$(J) = A$(I) + L$(J)
550  PRINT L$(J)
560 I = I + 1
570  GOTO 200
580 :
590 :
700  REM  **--CHECK COLUMN C VALUES--**
710  IF A$(I) = "000" THEN J = J + 1
720  IF A$(I) = "100" THEN J = J + 1
730  IF A$(I) = "101" THEN J = J + 1
740  IF A$(I) = "110" THEN J = J + 1
750  IF A$(I) = "111" THEN J = J + 1
760  IF A$(I) = "001" THEN 900
770  IF A$(I) = "010" THEN 900
780  IF A$(I) = "011" THEN 900
790 L$(J - 1) = "00" + L$(J - 1)
800  PRINT L$(J - 1)
810 L$(J) = A$(I) + L$(J)
820 I = 2
830  PRINT L$(J)
840  GOTO 200
850 :
860 :
900  REM  **--NO PLOT COL.C VALUES--**
910 L$(J) = A$(I) + L$(J)
920  PRINT L$(J)
930 J = J + 1
940 I = 1
950  GOTO 200
960 :
970 :
1000  REM  **--SHAPE FINISHED, NOW CONVERT--**
1010 L$(J) = "00000000" + L$(J)
1020 L$(J) = RIGHT$ (L$(J),8)
1030  PRINT L$(J)
1040 J = J + 1
1050 L$(J) = "00000000"
1060  FOR Q = 1 TO J
```

```
1070 L$(Q) = "00000000" + L$(Q)
1080  PRINT L$(Q)
1090 L$(Q) =  RIGHT$ (L$(Q),8)
1100 H$(Q) =  LEFT$ (L$(Q),4)
1110 W$(Q) =  RIGHT$ (L$(Q),4)
1120  PRINT L$(Q),H$(Q),W$(Q)
1130 TP$ = H$(Q): GOSUB 5000:HX$(Q) = TP$
1140 TP$ = W$(Q); GOSUB 5000:HX$(Q) = HX$(Q) + TP$
1150  PRINT HX$(Q)
1160  NEXT Q
1170 :
1180 :
1500  REM  **--DEFINE SHAPE TABLE--**
1510  POKE 24576,1: POKE 24577,0: POKE 24578,4: POKE 24579,0: REM  START
     OF TABLE
1520  POKE 232,0: POKE 233,96
1530  FOR Q = 1 TO J
1540  PRINT HX$(Q),
1550 T2$ =  LEFT$ (HX$(Q),1): GOSUB 6000:T =  VAL (T2$)
1560 T = T * 16
1570 T2$ =  RIGHT$ (HX$(Q),1): GOSUB 6000:T2 =  VAL (T2$)
1580 T = T + T2
1590  PRINT T
1600  POKE 24579 + Q,T
1610  NEXT Q
1620 :
1630 :
2000  REM  **--PAUSE--**
2010  INPUT "";NUL$
2020 :
2030 :
3000  REM  **--DRAW SHAPE--**
3010  HGR
3020  HCOLOR= 3
3030  SCALE= 1
3040  ROT= 0
3050  DRAW 1 AT 140,80
3060 :
3070 :
4000  REM  **--END--**
4010  END
4020 :
4030 :
5000  REM  **--CONVERT TO HEXADECIMAL--**
5010  IF TP$ = "0000" THEN TP$ = "0"
5020  IF TP$ = "0001" THEN TP$ = "1"
5030  IF TP$ = "0010" THEN TP$ = "2"
5040  IF TP$ = "0011" THEN TP$ = "3"
```

```
5050  IF TP$ = "0100" THEN TP$ = "4"
5060  IF TP$ = "0101" THEN TP$ = "5"
5070  IF TP$ = "0110" THEN TP$ = "6"
5080  IF TP$ = "0111" THEN TP$ = "7"
5090  IF TP$ = "1000" THEN TP$ = "8"
5100  IF TP$ = "1001" THEN TP$ = "9"
5110  IF TP$ = "1010" THEN TP$ = "A"
5120  IF TP$ = "1011" THEN TP$ = "B"
5130  IF TP$ = "1100" THEN TP$ = "C"
5140  IF TP$ = "1101" THEN TP$ = "D"
5150  IF TP$ = "1110" THEN TP$ = "E"
5160  IF TP$ = "1111" THEN TP$ = "F"
5170  RETURN
5180 :
5190 :
6000  REM    **--CONVERT TO DECIMAL--**
6010  IF T2$ = "0" THEN T2$ = "0"
6020  IF T2$ = "1" THEN T2$ = "1"
6030  IF T2$ = "2" THEN T2$ = "2"
6040  IF T2$ = "3" THEN T2$ = "3"
6050  IF T2$ = "4" THEN T2$ = "4"
6060  IF T2$ = "5" THEN T2$ = "5"
6070  IF T2$ = "6" THEN T2$ = "6"
6080  IF T2$ = "7" THEN T2$ = "7"
6090  IF T2$ = "8" THEN T2$ = "8"
6100  IF T2$ = "9" THEN T2$ = "9"
6110  IF T2$ = "A" THEN T2$ = "10"
6120  IF T2$ = "B" THEN T2$ = "11"
6130  IF T2$ = "C" THEN T2$ = "12
6140  IF T2$ = "D" THEN T2$ = "13"
6150  IF T2$ = "E" THEN T2$ = "14"
6160  IF T2$ = "F" THEN T2$ = "15"
6170  RETURN
```

Animated Truck

```
100  REM  ***--ANIMATED.TRUCK--***
110 :
120 :
130  PRINT  CHR$ (4);"BLOAD TRUCK.TABLE"
140  POKE 232,0: POKE 233,96
150  SCALE= 3: ROT= 0
160  REM   **--INITIALIZE HI-RES GRAPHICS--**
170  HGR2
180  HCOLOR= 3: REM  SET COLOR TO WHITE
190 CC = 250:CR = 50
```

```
200 C1 = 250:R1 = 50
210 :
220 :
230  REM  **--INITIAL DRAW--**
240  DRAW 1 AT CC,CR
250  HGR : POKE  - 16302,0: DRAW 1 AT CC,CR
260 :
270 :
1000  FOR K = 1 TO 50
1010 :
1020  REM  *--ERASE FROM PAGE 2--*
1030  POKE 230,64: REM  DRAW ON PAGE 2
1040  XDRAW 1 AT CC,CR
1050 :
1060  REM  *--DRAW ON PAGE 2--*
1070 C2 = C1 - 2:R2 = R1
1080  DRAW 1 AT C2,R2
1090  POKE  - 16299,0: REM  SWITCH DISPLAY TO PAGE 2
1100 :
1110 CC = C1:CR = R1: REM   RESET CURRENT POSITIONS
1120 :
1130 :
1140  REM    *--ERASE FROM PAGE 1--*
1150  POKE 230,32: REM    DRAW ON PAGE 1
1160  XDRAW 1 AT CC,CR
1170 :
1180  REM    *--DRAW ON PAGE 1--*
1190 C1 = C2 - 2:R1 = R2
1200  DRAW 1 AT C1,R1: GOTO 1210
1210  POKE  - 16300,0: REM    SWITCH DISPLAY TO PAGE 1
1220 :
1230 CC = C2:CR = R2: REM    RESET CURRENT POSITIONS
1240  NEXT K
1250 :
1260 :
5000  REM  **--END--**
5010  INPUT "";L$
5020  TEXT
5030  END
```

Animated Plane

```
100  REM ***--ANIMATED.PLANE--***
110 :
120 :
130  PRINT  CHR$ (4);"BLOAD AIRPLANE.TABLE"
```

```
140  POKE 232,0: POKE 233,96
150  SCALE= 1: ROT= 0
160  REM   **--INITIALIZE HI-RES GRAPHICS--**
170  HGR2
180  HCOLOR= 3: REM  SET COLOR TO WHITE
190 CC = 25:CR = 50
200 C1 = 25:R1 = 50
210 :
220 :
230  REM    **--INITIAL DRAW--**
240  XDRAW 1 AT CC,CR
250  HGR : POKE  - 16302,0: XDRAW 1 AT CC,CR
260 :
270 :
1000  FOR K = 1 TO 50
1010 :
1020  REM  *--ERASE FROM PAGE 2--*
1030  POKE 230,64: REM   XDRAW ON PAGE 2
1040  XDRAW 1 AT CC,CR
1050 :
1060  REM   *--DRAW ON PAGE 2--*
1070 C2 = C1 + 2:R2 = R1
1080  XDRAW 1 AT C2,R2
1090  POKE  - 16299,0: REM  SWITCH DISPLAY TO PAGE 2
1100 :
1110 CC = C1:CR = R1: REM   RESET CURRENT POSITIONS
1120 :
1130 :
1140  REM    *--ERASE FROM PAGE 1--*
1150  POKE 230,32: REM    XDRAW ON PAGE 1
1160  XDRAW 1 AT CC,CR
1170 :
1180  REM     *--XDRAW ON PAGE 1--*
1190 C1 = C2 + 2:R1 = R2
1200  XDRAW 1 AT C1,R1: GOTO 1210
1210  POKE  - 16300,0: REM    SWITCH DISPLAY TO PAGE 1
1220 :
1230 CC = C2:CR = R2: REM    RESET CURRENT POSITIONS
1240  NEXT K
1250 :
1260 :
5000  REM  **--END--**
5010  INPUT "";L$
5020  TEXT
5030  END
```

Animated Truck and Plane

```
100  REM  ***--ANIMATE.TRK.PLN--***
110 :
120 :
130  PRINT  CHR$ (4);"BLOAD TRUCK.PLANE.TBL"
140  SCALE= 1: ROT= 0
150  REM   **--INITIALIZE HI-RES GRAPHICS--**
160  HGR2
170  HCOLOR= 3
180 CC = 26:CR = 50
190 :
200 :
210  REM  **--INITIAL DRAW--**
220  XDRAW 1 AT CC,CR: XDRAW 2 AT 250 - CC,CR + 100
230  HGR : POKE  - 16302,0: XDRAW 1 AT CC,CR: XDRAW 2 AT 250 - CC,CR +
    100
240 C1 = 26:R1 = 50
250 :
1000  FOR K = 1 TO 50
1010 :
1020  REM  *--ERASE FROM PAGE 2--*
1030  POKE 230,64: REM   DRAW ON PAGE 2
1040  XDRAW 1 AT CC,CR
1050  XDRAW 2 AT 250 - CC,CR + 100
1060 :
1070 :
1080  REM   *--DRAW ON PAGE 2--*
1090 C2 = C1 + 2:R2 = R1
1100  XDRAW 2 AT 250 - C2,R2 + 100
1110  XDRAW 1 AT C2,R2
1120  POKE  - 16299,0: REM  SWITCH DISPLAY TO PAGE 2
1130 :
1140 CC = C1:CR = R1: REM   RESET CURRENT POSITIONS
1150 :
1160 :
1170  REM    *--ERASE FROM PAGE 1--*
1180  POKE 230,32: REM     DRAW ON PAGE 1
1190  XDRAW 2 AT 250 - CC,CR + 100
1200  XDRAW 1 AT CC,CR
1210 :
1220 :
1230  REM     *--DRAW ON PAGE 1--*
1240 C1 = C2 + 2:R1 = R2
1250  XDRAW 2 AT 250 - C1,R1 + 100
1260  XDRAW 1 AT C1,R1
1270  POKE  - 16300,0: REM    SWITCH DISPLAY TO PAGE 1
1280 :
```

```
1290 CC = C2:CR = R2: REM    RESET CURRENT POSITIONS
1300  NEXT K
1310 :
1320 :
5000  REM  **--END--**
5010  INPUT "";L$
5020  TEXT
5030  END
```

Collision Demo

```
100  REM  ***--COLLISION--***
110 :
120 :
130  PRINT  CHR$ (4);"BLOAD TRK.PLN.EXP.TBL"
140  HGR : POKE  - 16302,0: REM  FULL SCREEN
150  SCALE= 1
160  ROT= 0
170 C = 60:R = 60
180 :
190 :
200  REM  **--OPTIONAL INPUT--**
210  REM INPUT "COLUMN #";C
220  REM  INPUT "ROW #";R
230 :
240 :
250  REM  **--DRAW BOTH SHAPES--**
260  FOR I = 1 TO 75
270 J = (R - 43) + I
280  IF J > R THEN J = R
290  XDRAW 1 AT C + I,J
300  XDRAW 2 AT (C + 100) - I,R
310 X =  PEEK (234)
320  IF X < 120 THEN 1000
330  XDRAW 2 AT (C + 100) - I,R
340  XDRAW 1 AT C + I,J
350  NEXT I
360 :
370 :
1000  REM  **--EXPLOSION--**
1010  FOR Z = 1 TO 3
1020  SCALE= Z
1030  XDRAW 3 AT (C + I + 5),R
1040  FOR Z = 1 TO 50: NEXT Q
1050  NEXT Z
1060  INPUT " ";NUL$
```

```
1070  TEXT
1080  END
```

Background for Animated Truck and Plane

```
100  REM  ***--ANT.TR.PL.BKGND--***
110 :
120 :
130  REM   **--DEFINE VARIABLES--**
140  REM  CC = CURRENT COLUMN POSITION
150  REM  CR = CURRENT ROW POSITION
160  REM  C1 = COLUMN POSITION ON PAGE 1
170  REM  R1 = ROW POSITION ON PAGE 1
180  REM  C2 = COLUMN POSITION ON PAGE 2
190  REM  R2 = ROW POSITION ON PAGE 2
200 CC = 0:CR = 50
210 C1 = 0:R1 = 50
220 :
230  REM  **--INITIALIZE SHAPE TABLE--**
240  PRINT  CHR$ (4);"BLOAD TRUCK2.PLN.TBL"
250  POKE 232,0: POKE 233,96
260  SCALE= 1: ROT= 0
270 :
280 :
290  REM  **--INITIALIZE HI-RES GRAPHICS PAGE 2--**
300  HGR2
310  HCOLOR= 1
320  HPLOT 0,0
330 CC = 0:CR = 50
340  CALL  - 3082
350 :
360  REM  **--DRAW BLACK BORDER AROUND MTS PAGE 2--**
370  HCOLOR= 0: HPLOT 0,120 TO 75,90 TO 85,92 TO 100,92 TO 175,130 TO 25
    0,120 TO 279,100: HPLOT 0,119 TO 74,89 TO 84,91 TO 99,91 TO 176,129 TO
    249,119 TO 279,99
380 :
390  REM  **--DRAW ROAD PAGE 2--**
400  FOR X = 155 TO 160: HPLOT 0,X TO 279,X: NEXT X
410 :
420  REM  **--DRAW BLUE SKY PAGE 2--**
430  HCOLOR= 6: FOR X = 0 TO 80: HPLOT 0,X TO 279,X: NEXT X
440  FOR X = 1 TO 50
450  HPLOT 0,119 - X TO 74 - X,89 - X TO 84 - X,91 - X TO 100 + X,91 - X
     TO 175 + X,129 - X TO 249 - X,119 - X TO 279,99 - X
460  NEXT X
470 :
```

```
480  REM  **--INITIAL DRAW ON PAGE 2--**
490  HCOLOR= 1
500  SCALE= 1: XDRAW 1 AT CC,CR: SCALE= 1: XDRAW 2 AT 279 - CC,CR + 100
510 :
520 :
530  REM  **--INITIALIZE HI-RES GRAPHICS PAGE 1--**
540  HGR : POKE  - 16302,0: HPLOT 0,0: CALL  - 3082: HCOLOR= 0
550 :
560  REM  **--DRAW BLACK BORDER AROUND MTS PAGE 1--**
570  HCOLOR= 0: HPLOT 0,120 TO 75,90 TO 85,92 TO 100,92 TO 175,130 TO 25
    0,120 TO 279,100: HPLOT 0,119 TO 74,89 TO 84,91 TO 99,91 TO 176,129 TO
    249,119 TO 279,99
580 :
590  REM  **--DRAW ROAD PAGE 1--**
600  FOR X = 155 TO 160: HPLOT 0,X TO 279,X: NEXT X
610 :
620  REM  **--DRAW BLUE SKY PAGE 1--**
630  HCOLOR= 6: FOR X = 0 TO 80: HPLOT 0,X TO 279,X: NEXT X
640  FOR X = 1 TO 50
650  HPLOT 0,119 - X TO 74 - X,89 - X TO 84 - X,91 - X TO 100 + X,91 - X
     TO 175 + X,129 - X TO 249 - X,119 - X TO 279,99 - X
660  NEXT X
670 :
680  REM  **--INITIAL DRAW ON PAGE 1--**
690  HCOLOR= 1
700  SCALE= 1: XDRAW 1 AT CC,CR: SCALE= 1: XDRAW 2 AT 279 - CC,CR + 100
710 :
720 :
1000  FOR K = 1 TO 500
1010 :
1020  REM  *--ERASE FROM PAGE 2--*
1030  POKE 230,64: REM   DRAW ON PAGE 2
1040  SCALE= 1: XDRAW 1 AT CC,CR: SCALE= 1
1050  XDRAW 2 AT 279 - CC,CR + 100
1060 :
1070  REM   *--DRAW ON PAGE 2--*
1080 C2 = C1 + 2:R2 = R1
1090  IF C2 > 279 THEN C2 = 0
1100  XDRAW 2 AT 279 - C2,R2 + 100
1110  SCALE= 1: XDRAW 1 AT C2,R2: SCALE= 1
1120  POKE  - 16299,0: REM  SWITCH DISPLAY TO PAGE 2
1130 :
1140 CC = C1:CR = R1: REM   RESET CURRENT POSITIONS
1150 :
1160 :
1170  REM    *--ERASE FROM PAGE 1--*
1180  POKE 230,32: REM     DRAW ON PAGE 1
1190  XDRAW 2 AT 279 - CC,CR + 100
```

```
1200  SCALE= 1: XDRAW 1 AT CC,CR: SCALE= 1
1210 :
1220  REM     *--DRAW ON PAGE 1--*
1230 C1 = C2 + 2:R1 = R2
1240  IF C1 > 279 THEN C1 = 0
1250  XDRAW 2 AT 279 - C1,R1 + 100
1260  SCALE= 1: XDRAW 1 AT C1,R1: SCALE= 1
1270  POKE  - 16300,0: REM    SWITCH DISPLAY TO PAGE 1
1280 :
1290 CC = C2:CR = R2: REM    RESET CURRENT POSITIONS
1300  NEXT K
1310 :
1320 :
5000  REM  **--END--**
5010  INPUT "";L$
5020  TEXT
5030  END
```

BLOAD Instructions for the Shape Tables

```
 BLOAD HORIZ.LNE.TABLE
*6000.600B
00/6000:01 00 04 00 2D 2D 2D 05 00 D0 CA CA-....---..PJJ

 BLOAD MULTI.SHP.TABLE
*6000.6036
00/6000:03 00 14 00 19 00 22 00 00 00 00 00 00 00 00 00-......".........
00/6010:00 00 00 00 2D 2D 2D 05 00 2D 2D 36 36 3F 3F 24-....---.--66??$
00/6020:24 00 2D 24 2C 36 2E 2D 3E 3F 36 3E 24 3C 3F 3F-$.-$,6.->?6>$<??
00/6030:26 27 25 2E 2D 00 C9-&'%.-.i

 BLOAD TRUCK.TABLE
*6000.6015
00/6000:01 00 04 00 D2 3F 37 27 27 2C 2D 24 2D 2D 2D 36-....R?7'',-$---6
00/6010:36 3F 3E 3C 00 2D-6?><.-

 BLOAD BLOAD AIRPLANE.TABLE
*6000.6027
00/6000:01 00 04 00 2D 24 24 24 2E 36 36 2E 2D 2D 3E 3F-....-$$$.66.-->?
00/6010:3F 36 36 3E 26 24 24 3C 3F 3F 3F 3E 26 3C 2C 24-?66>&$$<???>&<,$
00/6020:2E 2E 2D 2D 05 00 2E 2D-..--...-

 BLOAD TRUCK.PLANE.TBL
*6000.6039
00/6000:03 00 14 00 28 00 22 00 00 00 00 00 00 00 00 00-....(.".........
00/6010:00 00 00 00 2D 24 2C 36 2E 2D 3E 3F 36 3E 24 3C-....-$,6.->?6>$<
```

```
00/6020:3F 3F 26 27 25 2E 2D 00 D2 3F 37 27 27 2C 2D 24-??&'%.-.R?7'',-$
00/6030:2D 2D 2D 36 36 3F 3E 3C 00 D0----66?><.P

 BLOAD TRUCK2.PLN.TBL
*6000.606F
00/6000:02 00 14 00 28 00 00 00 00 00 00 00 00 00 00 00-....(...........
00/6010:00 00 00 00 2D 24 2C 36 2E 2D 3E 3F 36 3E 24 3C-....-$,6.->?6>$<
00/6020:3F 3F 26 27 25 2E 2D 00 92 52 09 25 F7 DB 1B 27-??&'%.-..Rw[.'
00/6030:35 E4 DB 2B 2D 2D 2D 2D 2D 2D 2D 3C 3F 3F 3F 3F-5d[+-------<????
00/6040:3F 3F 3F 2C 2D 2D 2D 2D 2D 2D 2D 3C 3F 3F 3F 3F-???,-------<????
00/6050:3F 3F 3F 4C 09 2D 2D 2D 2D 2D 25 3F 3F 3F 3F 3F-???L.-----%?????
00/6060:27 2D 2D 2D 2D 2D 25 3F 3F 3F 3F 3F 2F FF 00 FF-'-----%?????/...

 BLOAD TRK.PLN.EXP.TBL
*6000.6091
00/6000:03 00 14 00 28 00 70 00 00 00 00 00 00 00 00 00-....<.p.........
00/6010:00 00 00 00 2D 24 2C 36 2E 2D 3E 3F 36 3E 24 3C-....-$,6.->?6>$<
00/6020:3F 3F 26 27 25 2E 2D 00 92 52 09 25 F7 DB 1B 27-??&'%.-..R.%w[.'
00/6030:35 E4 DB 2B 2D 2D 2D 2D 2D 2D 2D 3C 3F 3F 3F 3F-5d[+-------<????
00/6040:3F 3F 3F 2C 2D 2D 2D 2D 2D 2D 2D 3C 3F 3F 3F 3F-???,-------<????
00/6050:3F 3F 3F 4C 09 2D 2D 2D 2D 2D 25 3F 3F 3F 3F 3F-???L.-----%?????
00/6060:27 2D 2D 2D 2D 2D 25 3F 3F 3F 3F 3F 2F FF 00 FF-'-----%?????/...
00/6070:36 F6 95 F9 DB E0 68 0D 38 2C 3C 9F 1F 0C 08 18-6v.y[`h.8,<.....
00/6080:08 4D 2A 68 B0 33 37 26 35 05 20 55 D6 8D 8D 04-.M*h037&5. UV...
00/6090:00 80-..

 BEOAD BLOAD LARGE.SQ.TABLE
*6000.602D
00/6000:01 00 04 00 36 36 36 36 36 36 36 36 36 36 3F 3F-....6666666666??
00/6010:3F 3F 3F 3F 3F 3F 3F 3F 24 24 24 24 24 24 24 24-????????$$$$$$$$
00/6020:24 24 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 00 DB-$$----------.[

 BLOAD I.TABLE
*6000.6010
00/6000:01 00 04 00 2D 2D 3F 36 36 36 36 3F 2D 2D 07 00-....--?6666?--..
00/6010:3F-?

 BLOAD SM.CIRCLE.TABLE
*6000.6013
00/6000:01 00 04 00 92 2D 28 28 20 1C 1C 1C BF 17 17 76-.....-(( ...?..v
00/6010:0E 0E 00 3F-...?

 BLOAD SOLID.CRL.TABLE
*6000.6035
00/6000:01 00 04 00 36 2E 24 24 24 AC 36 36 2E 20 24 15-....6.$$$,66. $.
00/6010:3E 3F 27 24 BC 36 36 3E 20 24 17 2E 2D 25 2C 2D->?'$<66> $..-%,-
00/6020:1C 3F 37 3F 67 2D 27 2D E5 37 36 36 3F 0E 2D 25-.?7?g-'-e766?.-%
00/6030:2F F5 37 27 00 2D-/u7'.-
```

```
 BLOAD ASCII.SHP.TABLE
*6000.62FF
00/6000:96 00 00 00 00 00 00 00 00 00 00 00 00 00 00-................
00/6010:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00-................
00/6020:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00-................
00/6030:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00-................
00/6040:E5 05 2E 01 33 01 3C 01 4F 01 5E 01 6B 01 7E 01-e...3.<.O.^.k.~.
00/6050:91 01 99 01 A1 01 B7 01 C0 01 C6 01 CB 01 DE 01-....!.7.@.F.K.^.
00/6060:E7 01 F9 01 0D 02 21 02 2B 02 35 02 4A 02 5E 02-g.y...!.+.5.J.^.
00/6070:66 02 73 02 87 02 9A 02 AE 02 B7 02 BF 02 C7 02-f.s.......7.?.G.
00/6080:DA 02 E8 02 F7 02 0B 03 17 03 2B 03 39 03 45 03-Z.h.w.....+.9.E.
00/6090:59 03 66 03 6F 03 83 03 93 03 9A 03 AA 03 B9 03-Y.f.o.......*.9.
00/60A0:C4 03 D0 03 DF 03 EE 03 FB 03 0F 04 18 04 22 04-D.P._.n.{.....".
00/60B0:37 04 43 04 4C 04 61 04 6A 04 72 04 7B 04 8E 04-7.C.L.a.j.r.{...
00/60C0:97 04 9B 04 A8 04 B5 04 C2 04 CF 04 DB 04 E6 04-....(.5.B.O.[.f.
00/60D0:F6 04 00 05 04 05 1A 05 27 05 2E 05 3C 05 46 05-v.......'...<.F.
00/60E0:50 05 5E 05 74 05 7D 05 89 05 9E 05 A8 05 B3 05-P.^.t.}.....(.3.
00/60F0:BF 05 CE 05 DA 05 00 00 00 00 00 00 00 00 00 00-?.N.Z...........
00/6100:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00-................
00/6110:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00-................
00/6120:00 00 00 00 00 00 00 00 00 00 00 00 00 00 31 36-..............16
00/6130:B6 4D 00 31 36 24 4C 36 26 24 06 00 31 36 36 26-6M.16$L6&$..166&
00/6140:2C 3F 2D 2D 37 26 24 24 34 3E 3F 2D 2D 07 00 09-,?--7&$$4>?--...
00/6150:36 36 36 3C 2F 2D 0C 1C 3F 1C 0C 2D 3D 00 35 27-666</-..?..-=.5'
00/6160:4D 11 1E 1E 1E 1E 4E 21 35 27 00 69 09 1A 1B 1F-M.....N!5'.i....
00/6170:6E 4D 1A 1B 3B 6A 0D 15 1B 1F 73 6D 15 00 49 0D-nM..;j....sm..I.
00/6180:1A 1B 1F 0A 4D 11 1B 1B 53 49 11 1B 1B 53 49 11-....M...SI...SI.
00/6190:00 09 1E 1E 36 0E 0E 1C 00 09 0E 0E 36 1E 1E 0C-....6.......6...
00/61A0:00 09 36 36 36 34 24 F4 1E 0C 0C 0C 0C 1E 1E 1C-..6664$t........
00/61B0:1C 0E 0E 0E 0E 1C 00 89 36 36 24 3F 2D 2D 07 00-........66$?--..
00/61C0:49 92 32 1E 0C 00 92 2D 2D 07 00 49 09 1A 1B 1B-I.2....--..I....
00/61D0:4A 49 1A 1B 1B 4A 49 1A 1B 3F 0A 6D 11 00 49 11-JI...JI..?.m..I.
00/61E0:1E 1E 1E 1E 1E 0C 00 29 75 36 36 1E 3F 1C 24 24-.......)u66.?.$$
00/61F0:0C 2D 0E 1E 1E 1E 1E 0C 00 09 4D 1A 1B 3F 4A 4D-.-........M..?JM
00/6200:1A 1B 1F 4A 4D 1A 1B 1F 0A 2D 0D 02 00 29 6D 1A-...JM....-...)m.
00/6210:1F 1B 4E 09 15 1B 3F 53 4D 11 1B 1B 33 2D 2D 15-..N...?SM...3--.
00/6220:00 2D 2D F6 1E 75 F6 3F 1C 0E 00 36 2E 2D 3D 24-.--v.uv?...6.-=$
00/6230:34 36 36 26 00 2D 2D 15 1B 1B 33 2D 6D 1A 1F 1B-466&.--...3-M...
00/6240:4A 09 15 3B 1B 73 2D 0D 02 00 29 6D 1A 1F 1B 6E-J..;.s-...)m...n
00/6250:49 1A 3B 3F 6E 09 15 3B 1B 73 2D 0D 02 00 2D 2D-I.;?n..;.s-...--
00/6260:F6 1E 1E 36 04 00 29 75 F6 3F 1E 76 2D 0C E4 3F-v..6..)uv?.v-.d?
00/6270:1C 64 00 29 6D 1A 1F 1B 6E 09 15 3B 3F 57 49 15-.d.)m...n..;?WI.
00/6280:3B 1B 73 2D 0D 02 00 49 09 1A 1B 3F 0A 6D 11 1B-;.s-...I...?.m..
00/6290:1B 53 6D 11 1B 3B 57 49 11 00 49 09 1A 1B 3F 0A-.Sm..;WI..I...?.
00/62A0:6D 11 1B 1B 53 6D 11 1B 3B 17 6D 09 02 00 49 1E-m...Sm..;.m...I.
00/62B0:1E 1E 0E 0E 0E 1C 00 12 2D 2D 16 3F 3F 05 00 71-........--.??..q
00/62C0:0E 0E 1E 1E 1E 0C 00 29 6D 1A 1F 1B 4E 09 15 1B-.......)m...N...
```

```
00/62D0:3F 53 4D 11 1B 1B 53 4D 11 00 29 75 36 1E 27 34-?SM...SM..)u6.'4
00/62E0:DE 24 34 36 76 2D 3D 00 90 0E 0E 36 36 24 3C 3F-^$46v-=....66$<?
00/62F0:37 36 24 24 0C 1E 00 2D 6D 1A 1F 3B 0A 4D 15 1B-76$$...-m..;.M..
00/6300:3F 57 4D 15 3B 1B 17 2D 6D 02 00 49 0E 1C 3F 1E-?WM.;..-m..I..?.
00/6310:36 36 0E 2D 0C 1E 00 2D 6D 1A 1F 3B 0A 4D 15 3B-66.-...-m..;.M.;
00/6320:1B 57 4D 15 3B 1B 17 2D 6D 02 00 2D 2D 3F 3F 36-.WM.;..-m.--??6
00/6330:2E 2D 3F 37 36 2D 2D 07 00 2D 2D 3F 3F 36 2E 2D-.-?76--..--??6.-
00/6340:3F 37 36 04 00 29 6D 1A 1F 1B 6E 49 1A 3F 1F 6E-?76..)m...nI.?.n
00/6350:09 15 3B 1B 73 2D 0D 02 00 36 36 36 24 2C 2D 25-..;.s-...666$,-%
00/6360:24 36 36 36 04 00 09 3D 2F 36 36 36 2F 3D 00 09-$666...=/666/=..
00/6370:2D 15 1B 1F 53 09 0D 1A 3B 1B 4A 69 1A 3B 1B 0E--...S...;.Ji.;..
00/6380:6D 11 00 36 36 36 24 2C 0C 0C 0C 1E 1E 1E 0E 0E-m..666$,........
00/6390:0E 14 00 36 36 36 2D 2D 07 00 36 36 36 24 24 24-...666--..666$$$
00/63A0:0E 0E 26 0C 0C 36 36 36 24 00 36 36 36 24 24 24-..&..666$.666$$$
00/63B0:76 0E 0E 0E 26 24 24 34 00 29 75 36 36 1E 3F 1C-v...&$$4.)u66.?.
00/63C0:24 24 0C 00 36 36 36 24 24 24 2D 75 F6 3F 37 00-$$..666$$$-uv?7.
00/63D0:29 75 36 DE 0E 0E 1C 0C 16 1F E7 24 24 4C 00 36-)u6^......g$$L.6
00/63E0:36 36 24 24 24 2D 75 F6 3F 0E 0E 0E 1C 00 29 75-66$$$-uv?.....)u
00/63F0:1C 3F 1E 76 2D 0E F6 3F 1C 0E 00 2D 2D 15 3B 3B-.?.v-.v?...--.;;
00/6400:73 69 11 1B 3B 53 69 11 1B 3B 53 2D 0D 02 00 36-si..;Si..;S-...6
00/6410:36 76 2D 0C 24 24 34 00 36 36 0E 0E 0C 0C 24 24-6v-.$$4.66....$$
00/6420:06 00 4D 29 1A 1F 1B 6E 09 15 3B 1B 33 0D 0D 15-..M)...n..;.3...
00/6430:3B 1F 37 4D 29 02 00 76 0E 0E 0E FE 1B 64 0C 0C-;.7M)..v... .d..
00/6440:0C 34 00 76 0E 36 26 24 0C 0C 34 00 2D 2D 15 3B-.4.v.6&$..4.--.;
00/6450:1B 73 09 0D 1A 1B 1F 0A 4D 11 3B 1B 33 2D 2D 15-.s......M.;.3--.
00/6460:00 2D 3F 36 36 36 2D 07 00 12 0E 0E 0E 0E 0E 07-.-?666-.........
00/6470:00 49 49 2D 36 36 36 3F 07 00 09 4D 1A 3B 3B 6A-.II-666?...M.;;j
00/6480:09 15 1B 1B 53 49 11 1B 1B 53 49 11 00 92 92 92-....SI...SI.....
00/6490:2F 2D 2D 2D 07 00 12 0E 0E 07 00 92 0C 75 0E 24-/---.........u.$
00/64A0:36 36 24 3E 1E E7 34 00 36 36 36 24 0C 0C 75 36-66$>.g4.666$..u6
00/64B0:1E E7 1C 04 00 92 0C 2D 0E 1C 3F 1E 36 0E 2D 0C-.g.....-..?.6.-.
00/64C0:1E 00 49 31 36 36 26 F7 E7 24 0C 2D 0E 36 00 92-..I166&wg$.-.6..
00/64D0:22 0C 2D 0E 3E 3F 37 0E 2D 07 00 92 2D 27 64 75-".-.>?7.-...-'du
00/64E0:1C F7 36 36 26 00 92 0C 2D 0E 36 26 F7 E7 24 36-.w66&...-.6&wg$6
00/64F0:0E 4D F6 3F 05 00 36 36 36 24 0C 0C 75 36 26 00-.Mv?..666$..u6&.
00/6500:89 16 36 26 00 12 49 29 1A 1B 1B 4A 29 15 3B 1B-..6&..I)...J).;.
00/6510:53 49 15 3B 1B 73 2D 0D 02 00 36 36 36 24 2D 0C-SI.;.s-...666$-.
00/6520:0C 1E 1E 0E 0E 1C 1C 00 89 36 36 36 04 00 92 36-.........666...6
00/6530:26 24 0C 0E 36 24 26 0C 0E 36 26 00 12 36 36 24-&$..6$&..6&..66$
00/6540:2C 0C 75 36 26 00 92 0C 2D 0E 36 1E 3F 1C 24 06-,.u6&...-.6.?.$.
00/6550:00 92 36 36 36 24 2C 0E 65 24 1C 3F 1E 06 00 12-..666$,.e$.?....
00/6560:49 09 1A 1F 3F 6A 29 15 3B 1F 73 6D 15 3B 1B 53-I...?j).;.sm.;.S
00/6570:49 15 15 00 12 36 36 24 2C 0C 75 1C 00 92 0C 2D-I....66$,.u....-
00/6580:3D 3F 1E 0E 2D 0E 1E 3F 2F 00 12 69 09 1A 1B 3F-=?..-..?/..i...?
00/6590:0E 4D 11 1B 1B 57 4D 15 1B 3F 53 49 11 00 12 36-.M...WM..?SI...6
00/65A0:76 65 0C 24 36 36 04 00 12 36 0E 0E 1C 0E 0C 0C-ve.$66...6......
00/65B0:24 06 00 12 36 36 0C 0C 0E 0E 24 24 26 06 00 12-$...66....$$&...
00/65C0:0E 0E 0E 0E 1C 1C 1E 1E 0C 0C 0C 0C 1E 00 12 36-..............6
```

```
00/65D0:76 65 0C 24 36 36 F6 3F 05 00 12 2D 2D 1E 1E 1E-ve.$66v?...--...
00/65E0:1E 2D 2D 07 00 92 92 92 00 00 00 00 00 00 00 00-.--.............
00/65F0:00 00 00 00 00 FF 00 FF 00 00 00 00 00 00 00 00................
```

CHAPTER 9

High-Resolution Graphs

Regardless of the kind of graph or the type of data you are working with, the creation and use of high-resolution graphs involve four steps.

1. You need to learn how to draw the different types of graphs that can be generated on your computer.
2. Since all graphs require some kind of labels, you must learn to use high-resolution character shapes to label the different types of graphs.
3. You must read in data from some source—usually from a disk drive. Instruction on this aspect of high-resolution graphs is given in depth in my book *Apple Files*, which fully explains the storage and retrieval of data, but I will include enough information and examples here that you should be able to accomplish this task.
4. The data must be translated into graphic information and plotted on the high-resolution screen.

I will discuss each of these steps in this chapter.

There are many different types of graphs. The most common are: bar, column, line, scatter, high/low, area, pie and deviation graphs. Less common are the more complex graphs: stacked column, percentage bar, and pictograph. I will describe some of these and how they are created. In addition I will show how certain techniques can be used to create virtually any graph. Figures 9.1 through 9.6 show the different types of graphs. Later in the chapter, I will explain different instructions necessary to create these graphs.

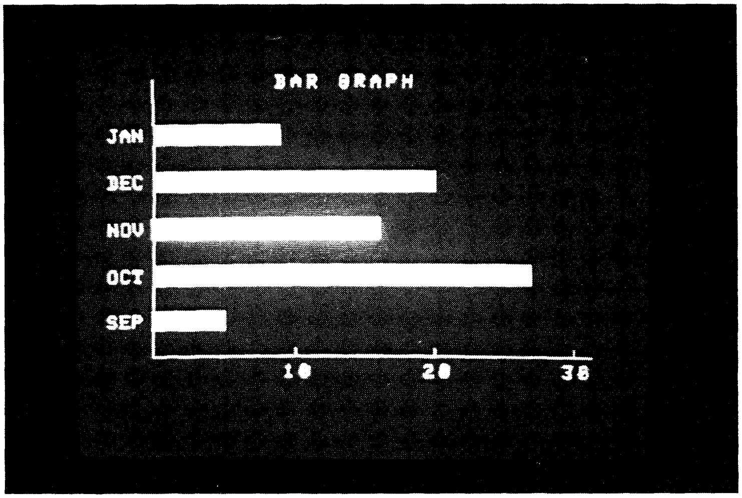


FIGURE 9.1. Bar graph.

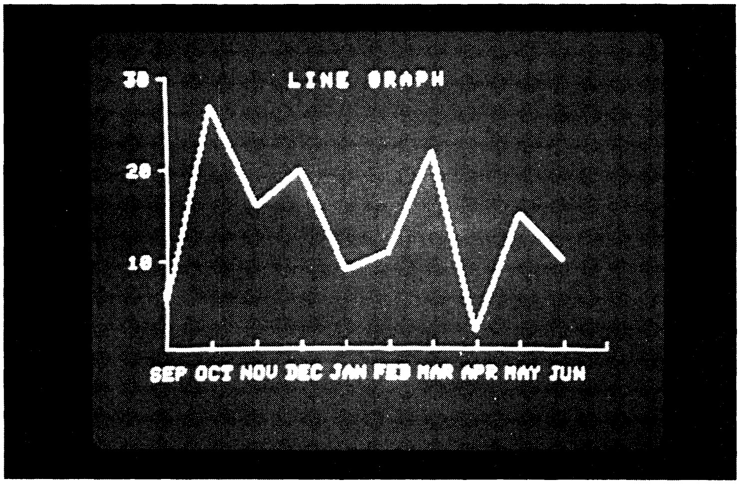


FIGURE 9.2. Line graph.

MEASUREMENT LINES

With the exception of the pie graph, all graphs use a left vertical line and at least one bottom horizontal line for their measurements. The first task then is a fairly simple one: draw both measurement lines. Type in the following:

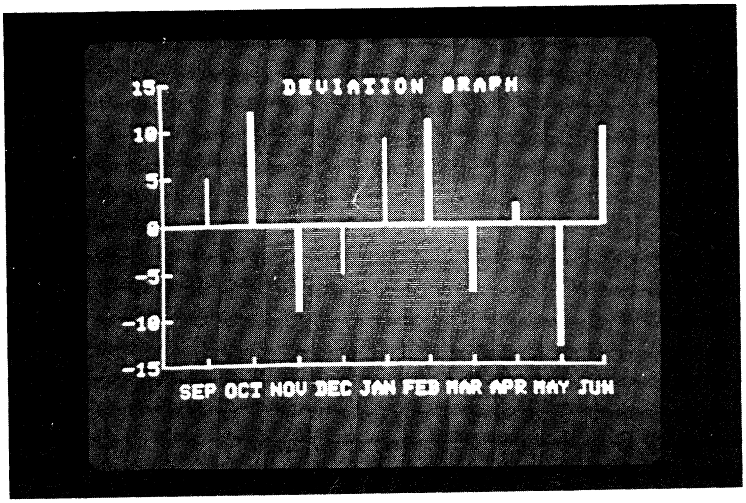


FIGURE 9.3. Deviation graph.

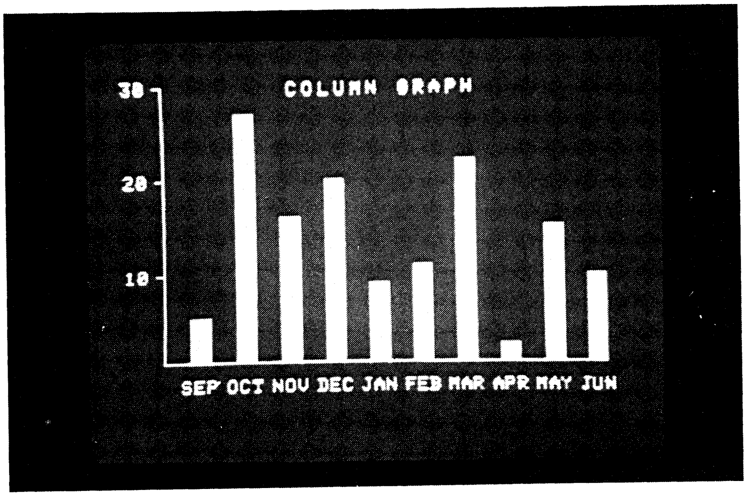


FIGURE 9.4. Column graph.

```
100 REM ***--DRAW.GRAPH--***
110:
120:
130 REM **--SET UP GRAPHICS--**
140 HGR
150 POKE -16302,0:REM FULL SCREEN GRAPHICS
160 HCOLOR = 5:REM BROWN
```

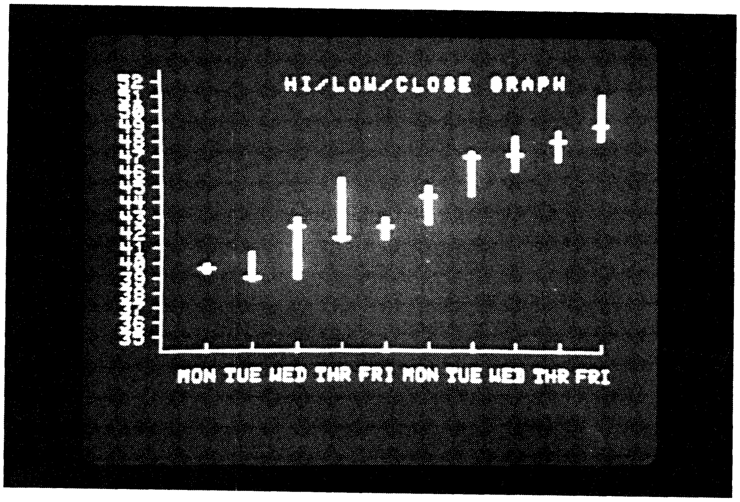


FIGURE 9.5. Hi/low/close graph.

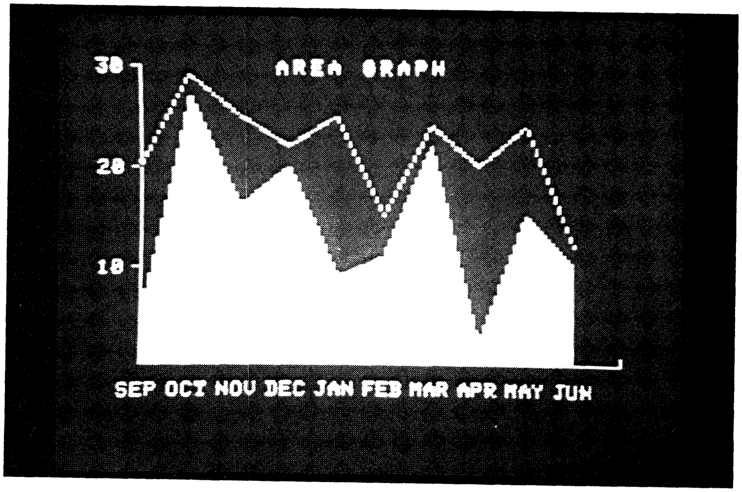


FIGURE 9.6. Area graph.

```
170 C = 25:REM COLUMN VALUE
180 ROW = 157:REM ROW VALUE
190:
195:
200 REM **--DRAW GRAPH LINES--**
210 HPLOT C,ROW-150 TO C,ROW:REM LEFT VERTICAL LINE
220 HPLOT C,ROW TO C+250,ROW:REM BOTTOM HORIZ.LINE
```

```
230:
240:
```

Instruction line 210 draws a vertical line on the left-hand side of the screen at the 25th column (C), beginning at row 7 (ROW − 150 where ROW = 157) and ending on row 157 (ROW). Instruction line 220 draws a horizontal line along the bottom of the screen on row 157 (ROW) beginning at the 25th column (C) and proceeding to the 275th column (C + 250).

The following code will complete the initial stage of this type of graph. Type

```
300 REM **--SHORT HORIZ.LINES--**
310 FOR I = ROW - 150 TO ROW - 50 STEP 50
320 HPLOT C,I TO C - 4,I
330 NEXT I
340:
350:
400 REM **--SHORT VERT.LINES--**
410 HCOLOR = 7:REM WHITE
420 FOR I = C + 25 TO C + 250 STEP 25
430 HPLOT I,ROW - 4 TO I,ROW
440 NEXT I
450:
460:
900 REM **--END--**
910 INPUT "";NUL$
920 TEXT
930 END
940:
950:
```

Instructions 310 to 330 tell the computer to draw three short horizontal lines on rows 7, 57 and 107. Each of these lines extends from the 25th column (C) to the 21st column (C − 4). These three short horizontal lines can be used as measurement lines to indicate the values of the points that will be plotted.

Instructions 410 to 440 tell the computer to draw short vertical lines on every 25th column. Each of these lines extends from the 153rd row (ROW − 4) to the 157th row (ROW). These short vertical lines can be used as measurement lines to indicate the values of the points that will be plotted. The change in color in line 410 is necessary so that every line will be displayed. Remember that, except for the color white, even-numbered colors are displayed in even-numbered columns and odd-numbered colors are displayed in odd-numbered columns. If you leave the HCOLOR setting at brown (5) only the odd-numbered columns will display lines (Fig. 9.7).

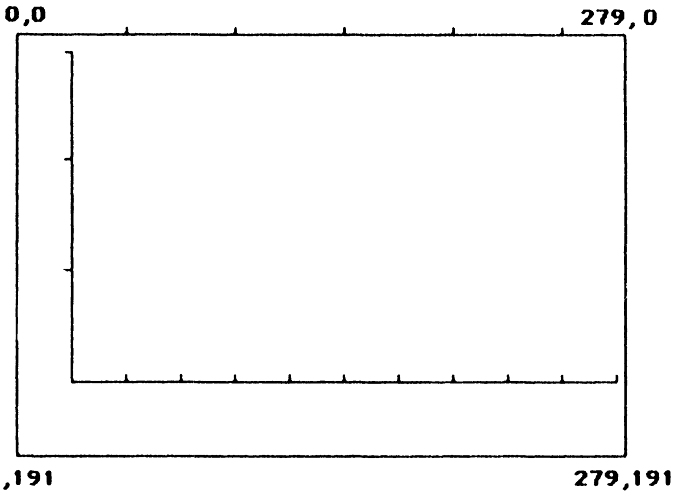


FIGURE 9.7. Graph lines.

The routine at 900 ends the program by allowing the user to press the RETURN key when he or she is finished viewing the graph (INPUT " ";NUL$). The computer switches from displaying High-Resolution Page 1 to displaying Text Page 1.

SCREEN POINTS AND DATA POINTS

The procedure used to establish the parameters for this graph can be used for most graphs. The area of the screen that will be used to plot points is established first. In this case, there are 150 possible vertical points (157 − 7) and 250 possible horizontal points (275 − 25). Further, the vertical points have been divided into three sections, with 50 points in each section, and the horizontal points have been divided into 10 sections, with 25 points in each section.

Before you label these sections, you need to establish the relationship between screen points and data points. The relationship may be one to one (i.e., one data value equals one screen point), but more often there is a different ratio between the two. In the above example, I have established the ratio of 5:1 (i.e., five vertical screen points equals one data value). In other words, there are five screen points between every unit of data value.

There are many ways to determine the ratio to be used, including allowing the computer to determine what the ratio should be. The STOCK.PRICE and STOCK-.VOLUME programs included at the end of this chapter demonstrate the technique of having the computer read in a certain amount of data and determine from that data what the ratio of screen points to data values should be. For the current example, I decided that no data value would exceed 30. In other words, 30 would be the maximum possible data

value. In specific instances, this could mean that no student could score more than 30 points on a quiz or that the maximum number of products that could be sold would be 30. It does not matter what the data value represents, just that no data value in this example can exceed 30. If I have a maximum of 30 data values and a maximum of 150 vertical screen points, then five vertical screen points can represent one data value (150 screen points ÷ 30 data values, or a 5:1 ratio).

If you establish that the bottom horizontal line will represent a zero data value, the first short horizontal line on the vertical axis will represent a data value of 10 (50 screen points ÷ 5). The second short horizontal line will have a data value of 20, and the top horizontal line will have a data value of 30. (Fig. 9.8*A*). The relationship between screen points and data values may seem confusing, and in fact it often is. Once the graph size is established and a ratio set, each point within the graph then has two different identifications: a definition by screen coordinates and a definition by graph coordinates (Fig. 9.8*B*).

LABELS

Now that the dimensions of the graph are established, and the horizontal and vertical boundaries are drawn you can turn your attention to providing labels for the graph.

Generally, text on a high-resolution graph must consist of ASCII character shapes. Therefore, before you can label any graph you must load (actually BLOAD) a shape table consisting of numbers and/or letters. The shape table I use in all the example programs in this chapter is the ASCII.SHP.TABLE from Chapter 8. If you have a different shape table

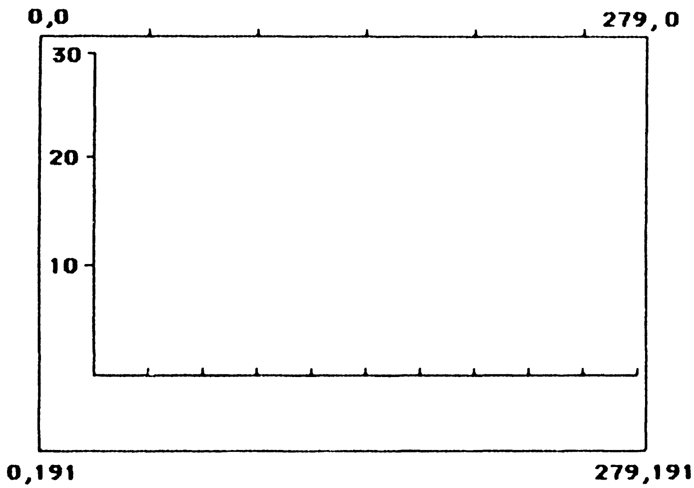


FIGURE 9.8. (*A*) Graph lines with left-side labels.

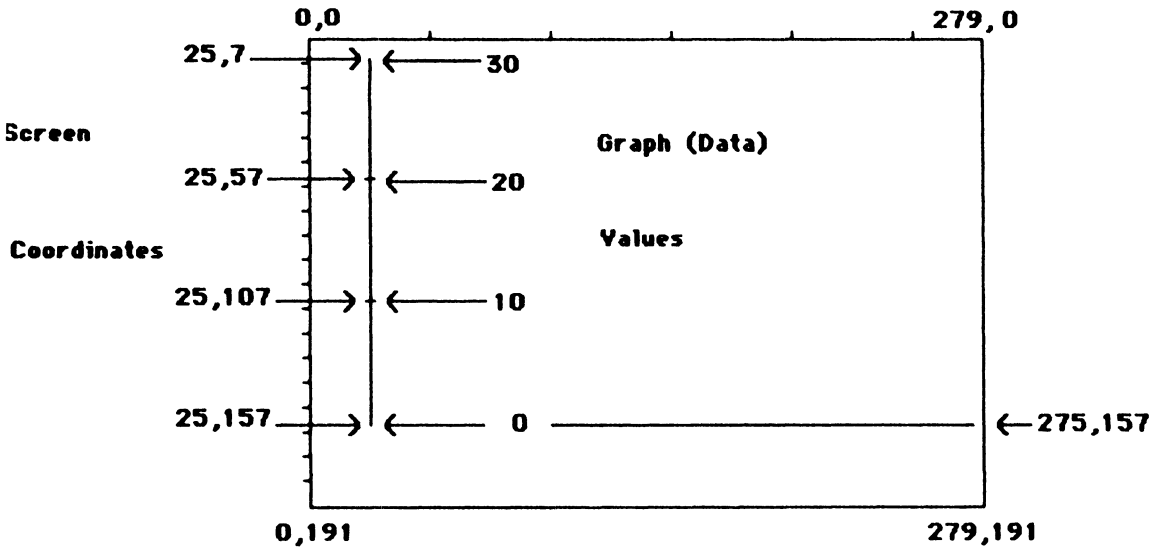


FIGURE 9.8. ***(B)*** **Screen coordinates and graph coordinates.**

you want to use, you can BLOAD it instead. If you are using the GRAPH.MENU program provided at the end of this chapter and want to use a different shape table, you should change the BLOAD instruction in line 156. If you want to use the ASCII.SHP.TABLE, but do not want to use the GRAPH.MENU program (you may want to try only one or two graphs), you should type the following instructions before running any of the graph programs:

```
BLOAD ASCII.SHP.TABLE {RETURN}
POKE 232,0 {RETURN}
POKE 233,96 {RETURN}
SCALE = 1 {RETURN}
ROT = 0 {RETURN}
HCOLOR = 3 {RETURN}
```

Remember, these instructions are necessary in order to use shapes in a shape table (for more information, review the instructions provided in Chapter 8).

Once you have the shape-table instructions out of the way, you can concentrate on how to use the shapes in labeling graphs. I have divided the labeling process into three different sections and placed all three sections into LABEL.ROUTINE, which has been added to all my graph programs.

Top Label

The first section to be explained is the top label subroutine.

```
7000 REM **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "LINE GRAPH"
7030 FOR T = 1 TO LEN (LABEL$)
7040 LB = ASC(MID$(LABEL,T,1))
7050 DRAW LB AT LC,LR
7060 LC = LC + 9
7070 NEXT T
7080:
7090:
```

The numeric variables LC (for Label Column) and LR (for Label Row) indicate the position for the start of the label that names the graph (in this case, LINE GRAPH). These variables contain screen-point values, not graph coordinates. In other words, the first character of the label will be at the 95th column on the 5th row. The string variable LABEL$ is assigned the title of the graph. String variables are used for both numbers and letters, because Applesoft BASIC contains useful string manipulation commands.

Beginning with the FOR instruction in line 7030, a loop is used to separate the contents of the string variable into individual characters and to assign the ASCII value for each character to the numeric variable LB. If you do not understand the LEFT$, MID$, and RIGHT$ statements, the best thing to do is to get a clear definition of them from a book devoted to teaching BASIC, and then practice using them. Essentially, they perform the functions for which they are named. The LEFT$ statement retrieves a specified number of characters beginning at the left side of a string variable. LEFT$(LABEL$,4) retrieves the first four characters of the string variable LABEL$; i.e., if LABEL$ = "LINE GRAPH", then LEFT$(LABEL$,4) = "LINE". The MID$ statement retrieves a specified number of characters from a specified position within a string variable. MID$(LABEL$,6,1) gets a single character at the 6th character in the string variable LABEL$; i.e., if LABEL$ = "LINE GRAPH", MID$(LABEL$,6,1) = "G".

The loop proceeds character by character through the contents of the string variable LABEL$. As each character is identified, the computer stores the ASCII code for that character in the numeric variable LB. The ASCII.SHP.TABLE is set up in such a way that the location of each character in the shape table matches the ASCII code for that character. In other words, the ASCII code for the letter A has the decimal value 65. Therefore, the shape for the capital letter A is the 65th shape in the shape table. It is very easy then, once the ASCII code for a character has been identified, for the computer to draw the shape for that character on the high-resolution screens. Instruction line 7050 draws the specified shape at the current location. That location is moved from left to right on the screen by

instruction line 7060, which adds the value of nine screen points to the current value of the column. Instruction line 7070 completes the loop with the NEXT instruction. The loop repeats for as many characters as there are in the string variable LABEL$. (If you are using a different shape table, you must know the shape number for each of the characters that may appear in a label.)

Bottom Label

The label for the bottom horizontal line follows a very similar routine.

```
8000 REM **--BOTTOM LABEL--**
8010 LC = C + 10
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040 FOR T = 1 TO LEN(LABEL$)STEP 3
8050 L1 = ASC (MID$(LABEL,T,1))
8060 L2 = ASC (MID$(LABEL,T + 1,1))
8070 L3 = ASC (MID$(LABEL,T + 2,1))
8080 DRAW L1 AT LC,LR:DRAW L2 AT LC + 7,LR:DRAW L3 AT LC + 1
8090 LC = LC + 25
8100 NEXT T
8110:
8120 INC = 0:REM RESET SCREEN POSITION TO 0
8130:
8140 RETURN
```

The only real difference between the two subroutines is in the FOR-NEXT loop. The loop uses the STEP parameter to proceed three characters at a time through the string variable LABEL$. Those three characters are drawn next to each other by advancing the column value seven positions for each character. The TOP LABEL subroutine advanced the column value nine positions for each character. After the three characters are drawn, the computer advances 25 column positions [8090] for the next set of three characters. All the characters are positioned 10 screen points below the bottom horizontal measurement line [8020].

Left-Side Label

The left-side label subroutine is a little more difficult.

```
6000 REM **--LEFT SIDE LABEL--**
6010 HCOLOR = 7:REM WHITE
```

```
6020 INC = 50:REM DISTANCE BETWEEN SCREEN POINTS
6030 LOW = 0:LOW = LOW + 10:REM DISTANCE BETWEEN GRAPH POINTS
6040 LOW$ = STR$(LOW):LN = LEN(LOW$)
6050 IF LN = 1 THEN D1 = ASC(LOW$):D2 = 32:D3 = 32
6060 IF LN = 2 THEN D1 = ASC(MID$(LOW$,2,1)):D2 = ASC
    (LEFT$(LOW$,1)):D3 = 32
6070 IF LN = 3 THEN D1 = ASC(MID$(LOW$,3,1)):D2 = ASC
    (MID$(LOW$,2,1)):D3 = ASC (LEFT$(LOW$,1))
6080 DRAW D1 AT C - 14,ROW - INC - 3
6090 DRAW D2 AT C - 21,ROW - INC - 3
6100 REM ETC. DRAW D3 AT C -28,ROW - INC - 3
6110 IF LOW = 30 THEN 7000:REM JUMP TO NEXT SUB-RTN
6120 LOW = LOW + 10:REM INCREASE DATA VALUE
6130 INC = INC + 50:REM INCREASE SCREEN POINT VALUE
6140 GOTO 6040:REM GO BACK FOR MORE
6150:
6160:
```

The idea of the subroutine is to label each of the short horizontal measurement lines. Since the maximum number possible is 30, the labels should include numbers between 0 and 30. The first measurement line should be labeled with the graph value of 10, the second measurement line should be labeled with the graph value of 20 and the top measurement line should be labeled with the graph value of 30. Knowing this, you could just instruct the computer to draw the individual shapes that represent each digit of each number at a specified point. But if you decided to change the maximum value from 30 to 75, the measurement lines would need to be relabeled using new instructions. With a little planning, this label subroutine can be written in such a way that the measurement lines can be labeled with any three-digit number. Increased flexibility, though, results in increased complexity. By changing the values in the instructions at 6020, 6030, 6110, 6120, and 6130, you can alter the quantities displayed at the measurement lines. These instructions redefine the ratio that exists between screen points and graph points.

For example, if the maximum graph value were 75 instead of 30, the ratio of screen points to graph points would be 2:1; i.e., two screen points for each graph point (150 screen points ÷ 75 graph points). In this example, only lines 6030, 6120 and 6110 would need to be changed. Since there are still three measurement lines, the distance between each line would be 25 instead of 10: the maximum number of graph points (75) divided by the number of measurement lines (3). Therefore, the value of 10 in lines 6030 and 6120 needs to be changed to 25, so that the values on the measurement lines would read 25, 50 and 75 instead of 10, 20 and 30. Since the screen location of the measurement lines remain the same, the increment value in 6020 and 6130 remains the same also. The value of 30 in instruction line 6110 must be changed to indicate the new maximum value, 75. Changing the values in three lines changes the numbers displayed next to each of the measurement lines and results

in the redefinition of the screen. The ease with which this redefinition is accomplished is the reason for developing a subroutine that does more than create specific labels.

Once the instructions at 6020, 6030, 6110, 6120 and 6130 are understood as the lines that increase the graph points and screen points, the rest of the subroutine can be understood in much the same way as the other two subroutines. The numeric value of LOW is converted to a string variable [6040] and that string variable is separated into three digits, D1, D2 and D3 [6050, 6060 and 6070]. The shapes corresponding to the ASCII code for those digits are then drawn at the correct location [6080, 6090 and 6100 if necessary].

There are a couple of things in this subroutine that need additional explanation. If there is only one digit in the number, the other two digits must be set to equal the value of the ASCII code for the space character—decimal value 32. If you don't assign the value of the space character to those digits, an error will occur. The subroutine could have been written in such a way as to avoid the assignment of the space character to the extra digits, but the subroutine would have been more complex and not as easily understandable as a result.

The placement of the shapes in the DRAW statement also needs additional explanation. The subroutine draws characters from right to left, and therefore the number subtracted from the column value increases with each character drawn.

The value subtracted from the ROW value in each of the DRAW statements is the same (ROW − INC − 3). This subtracts the distance between screen points (INC) and subtracts the constant value of 3 as well to position the measurement line in the approximate middle of the character shape.

I have purposely placed the DRAW D3.... statement in a REM statement, because the graph measurement line in this example extends to the left of the vertical line, taking up some of the room available for characters. If the measurement line extends to the right of the vertical line, then all three characters can easily fit within the first 25 columns.

The three label subroutines, added to the DRAW.GRAPH program, should produce what is shown in Figure 9.9.

DATA VALUES

After these subroutines are added to the DRAW.GRAPH program, the only thing left to do is to translate data values into screen positions. But before you can do any translating, you need some data values. Data values ordinarily come from three sources (1) a disk file, (2) user input, or (3) data statements included in the program. For this example we will use data statements.

```
1000 REM **--DATA VALUES--**
1010 DATA 5,27,16,20,9,11,22,2,15,10,0
1020:
1030:
```

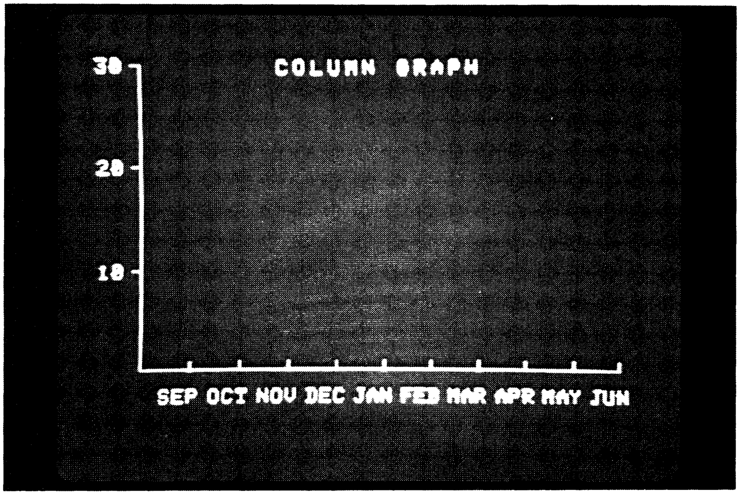


FIGURE 9.9 Graph lines and all labels.

These data values can be any numbers between 0 and 30, since I defined the graph as having a maximum data value of 30. If you have altered the left-side label subroutine so that the maximum data value can be 75, then the numbers in line 1010 can be any value between 0 and 75.

You now have data values and can proceed to translate those values into plotted positions on the high-resolution screen with the following routine:

```
600 REM **--PLOT DATA--**
610 HCOLOR = 3:REM WHITE
620 RTIO = 5:REM 5 SCREEN POINTS = 1 DATA POINT
630 READ DTPT:REM GET DATA POINT
640 IF DTPT = 0 THEN 900:REM GRAPH FINISHED
650 TPROW = ROW - (DTPT * RTIO):REM TOP ROW = VALUE OF DATA
660 HPLOT C + 25 + INC,TPROW
670 INC = INC + 25:REM INCREASE SCREEN POSITION FOR NEXT GRAPH
   POINT
680 GOTO 630:REM GO BACK FOR MORE POINTS
690:
695:
```

If you want to see a 75-point maximum, then the value in line 620 must be changed. 150 screen points divided by 75 graph points equals two. Therefore, the ratio necessary in line 620 is a value of two not a value of five.

One small additional routine must be added to the program before it can be run.

```
500 REM **--LABEL ROUTINE--**
510 GOSUB 6000:REM LABEL ROUTINES
520:
530:
```

Two lines should be changed to reflect the true nature of the program. Line 100 should read:

```
100 REM ***--SCAT.30.GRAPH--***
```

and line 7020 should be:

```
7020 LABEL$ = "SCATTER GRAPH"
```

After entering all of the above code, you should save the program by typing

```
SAVE SCAT.30.GRAPH {RETURN}
```

If you have used the maximum graph value of 75, then the program should be saved as SCAT.75.GRAPH. In either case, once the program is saved, and the proper shape-table instructions have been given to the computer (see Chapter 8), you can run the program. You should see a screen display similar to that shown in Figure 9.10. (If you have used the 75-point maximum, the points will be closer to the base line.)

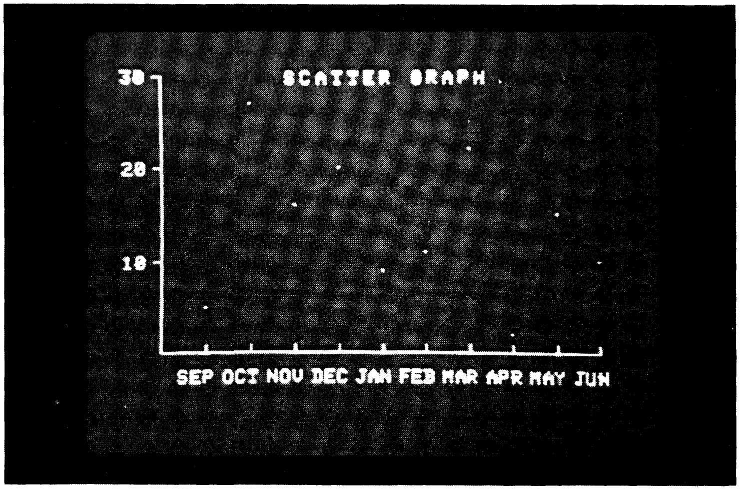


FIGURE 9.10. Scatter graph.

LINE GRAPH

By changing a few lines in the program, we can change the scatter graph into a line graph. Make the following additions and changes:

```
100 REM ***--LINE.30.GRAPH--***
655 IF INC = 0 THEN HPLOT C,TPROW:GOTO 670
660 HPLOT TO C + INC,TPROW
7020 LABEL$ = "LINE GRAPH"
8010 LC = C - 8
```

You should immediately save this new line-graph program:

```
SAVE LINE.30.GRAPH {RETURN}
```

If you want to see a line graph with the maximum value of 75, make the same changes you made in the SCAT.30.GRAPH program:

```
620 RTIO = 2:REM 2 SCREEN POINTS = 1 DATA POINT
6030 LOW = 0:LOW = LOW + 25:REM DISTANCE BETWEEN GRAPH POINTS
6110 IF LOW = 75 THEN 7000: REM JUMP TO NEXT SUB-RTN
6125 LOW = LOW + 25 REM INCREASE DATA VALUE
```

These changes result in a line-graph program with a maximum data value of 75. You should save this line-graph program under the name LINE.75.GRAPH:

```
SAVE LINE.75.GRAPH {RETURN}
```

As you can see, the change from a scatter graph to a line graph (Fig. 9.11) is minimal, yet the change produces a significantly different graph. The change essentially involves connecting the dots. In addition, for appearance' sake, plotting actually begins on the vertical line (C) rather than at the first short vertical line (C + 25). The first point must be plotted with the HPLOT statement not the HPLOT TO command. It is necessary therefore to add instruction line 655, which plots the first point correctly.

DEVIATION GRAPH

A deviation graph can also be created with relatively few additions and changes to the graph program. The concept of a deviation graph is that some values are above a certain point and other values are below a certain point. For the deviation graph example, the maximum

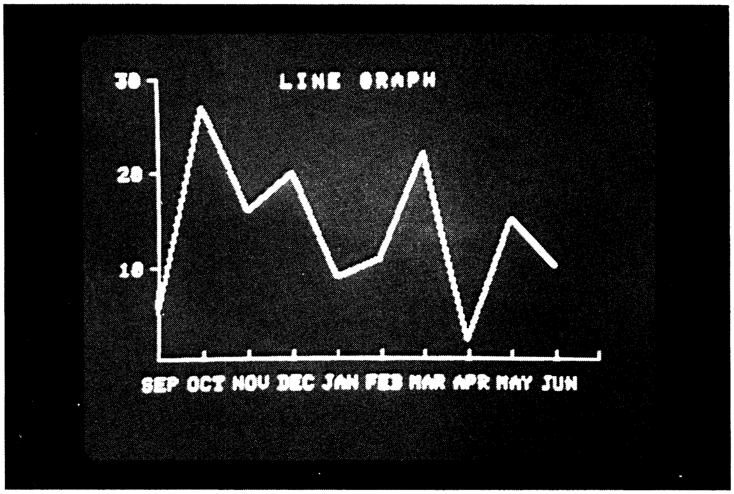


FIGURE 9.11. Line graph.

positive data value is 15 and the maximum negative data value is −15. We still have a 5:1 ratio, since there are still 150 screen points and a total of 30 graph points. Other than the plot data routine at 600, which is given below, there is only one additional instruction line:

```
225 HCOLOR = 7:HPLOT C,ROW - 75 TO C + 250,ROW - 75
```

This instruction draws another horizontal line in the middle of the graph area. That line is the deviation line, the line above and below which the points will be plotted.

Several lines must be changed as follows:

```
100 REM ***--DEVIT.15.GRAPH--***
310 FOR I = ROW - 150 TO ROW - 25 STEP 25
320 HPLOT C,I TO C + 4,I
1000 DATA 5,12,-9,-5,9,11,-7,2,-13,10,0
6020 INC = 0
6030 LOW = 0:LOW = LOW - 15
6080 DRAW D1 AT C - 7,ROW - INC - 3
6090 DRAW D2 AT C - 14,ROW - INC -3
6100 DRAW D3 AT C - 21,ROW - INC - 3
6110 IF LOW = 15 THEN 7000
6120 LOW = LOW + 5
6130 INC = INC + 25
7020 LABEL$ = "DEVIATION GRAPH"
```

The plot data routine requires a number of changes and additions:

```
610 INC = 25
650 TPROW = (ROW - 75) - (DTPT *RTIO)
655 IF DTPT > 0 THEN HCOLOR = 6
657 IF DTPT < 0 THEN HCOLOR = 5
660 HPLOT C + INC,ROW - 75 TO C + INC,TPROW
665 HPLOT C + INC + 1,ROW - 75 TO C + INC + 1,TPROW
666 HPLOT C + INC - 1, ROW - 75 TO C + INC - 1,TPROW
```

A total of 21 lines need to be changed or added to convert the SCAT.30.GRAPH or LINE.30.GRAPH program to a deviation graph (Fig. 9.12). I left out the REM portions of lines that were changed to reduce the time required to make the changes. If you are uncertain as to the purpose of those lines, add the remarks back in.

At the end of this chapter, I have included programs for bar graphs, column graphs, area graphs, and hi/low/close graphs, all of which are basically modifications of the graph programs already presented. The main differences occur in the plot data routine, but other routines or subroutines may also have changes that need to be made.

I have also included programs for a pie graph and a percent-distribution graph that can be used by teachers to plot grades. These programs follow some of the same principles, but require too many different routines for them to be modifications of the basic graph program. The remarks in both programs should greatly aid in understanding the techniques used to create them. The GRADE.GRAPH program (demonstrating percent distribution) produces additional information that is also displayed on the high-resolution screen with ASCII shapes from the ASCII.SHP.TABLE (Fig. 9.13). It is part of a complete grading

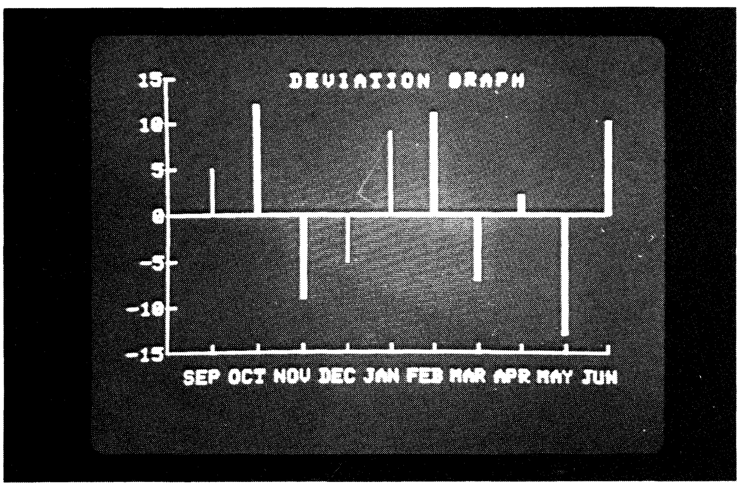


FIGURE 9.12. Deviation graph.

system that I developed and have made available to teachers at all educational levels. Readers interested in obtaining the complete program can get ordering information from the disk order page in the front of this book.

PIE GRAPH

The pie-graph program presents a number of problems. Certain colors do not mix well next to each other. The graph routine, except on the IIGS, is a slow routine, and the label routine is specific to the example program. I included this program, despite these problems, because it does provide an example of the problems involved in creating a generalized pie-graph program (Figs. 9.14 and 9.15).

I have also included two programs from a Stock Market System that produce graphic display of both a stock's price history and its volume history (Figs. 9.16 and 9.17). These programs demonstrate the techniques involved in obtaining data from a source other than DATA statements inside a program. The data used in the GRAPH.PRICE and GRAPH.VOLUME programs are stored on diskette in a random-access file called STOCKS. I strongly encourage readers interested in translating data stored on diskettes to study the techniques presented in these programs. These two programs demonstrate one of

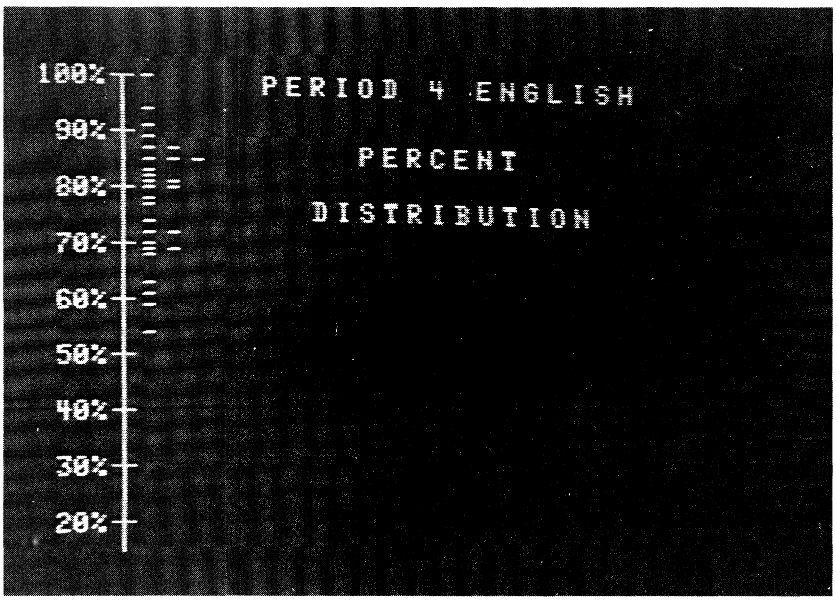


FIGURE 9.13. GRADE.GRAPH program.

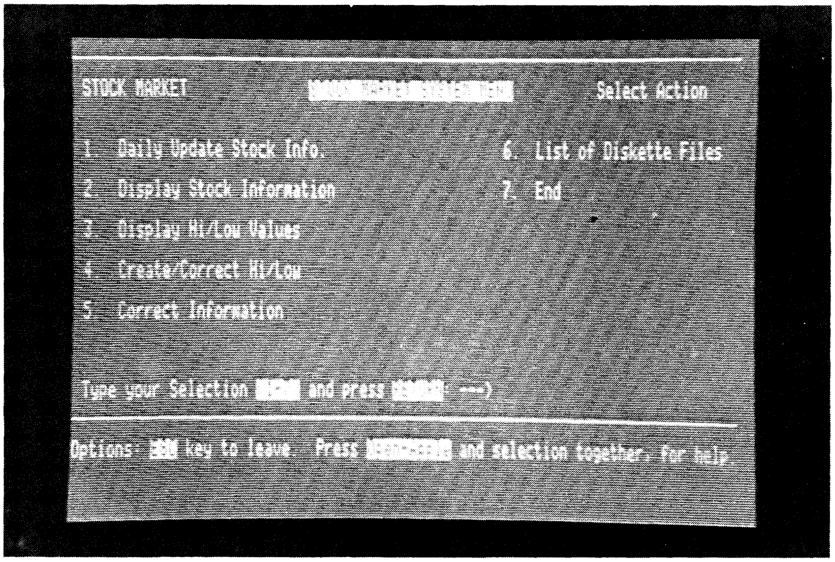


FIGURE 9.14. Stock Market System menu.

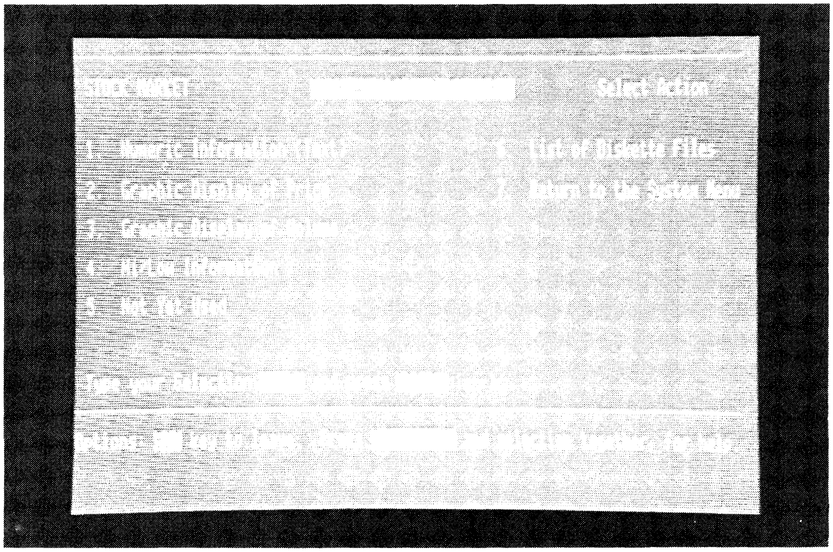


FIGURE 9.15. Display Stock Information menu.

the ways by which unknown data can be read, scaled, plotted, and labeled on the high-resolution screens. Readers interested in obtaining the complete Stock Market System should refer to the disk order page in the front of this book.

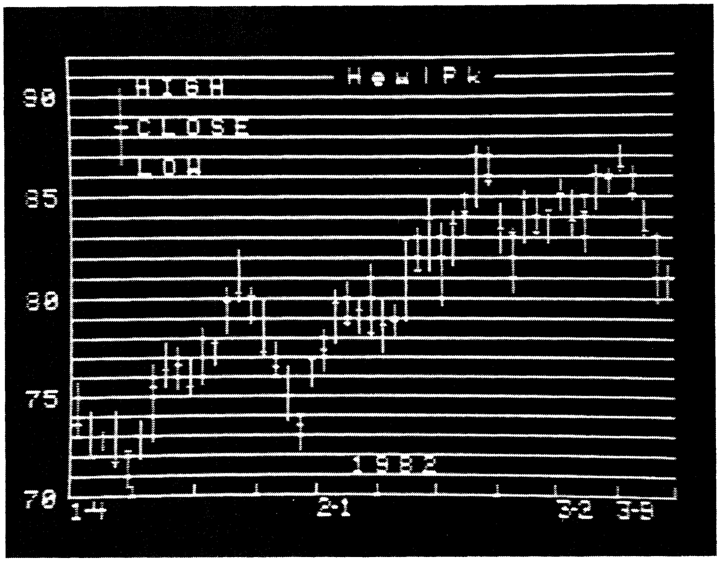


FIGURE 9.16. Hi/low/close stock graph.

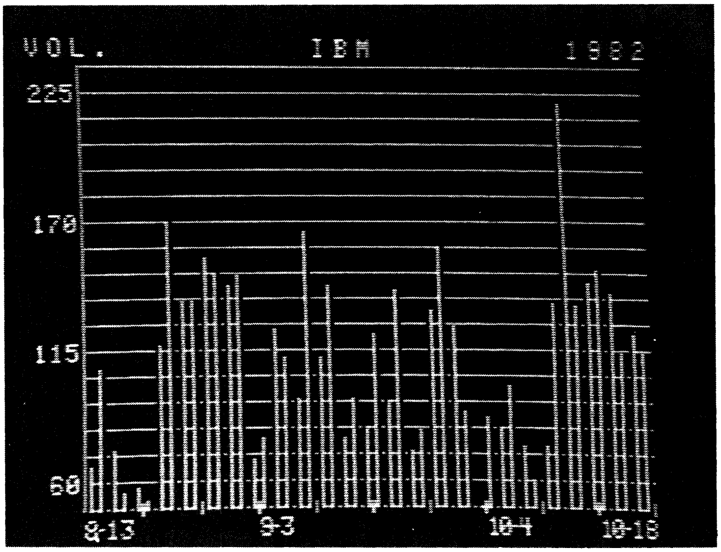


FIGURE 9.17. Stock volume graph.

Finally, I have included a GRAPH.MENU program that allows you to rapidly switch between the different graphic programs included in this chapter. A lot of effort has gone into

the creation and documentation of these programs. Once again, I encourage you to read through the listings of each program even if you do not intend to type in the listings. Although there is much more that could be written concerning high-resolution graphics and its many applications, this chapter concludes the study of high-resolution graphics. In the next chapter, I will discuss IIGS super high-resolution graphics and techniques that can be used to access super hi-res from Applesoft BASIC.

REVIEW QUESTIONS

1. Once a graph size is established and a ratio is set, each point within the graph then has two different identifications. What are they?
2. Text on a high resolution graph must consist of ______________.
3. True or False? In the ASCII.SHP.TABLE, the shape index number of each character corresponds to the actual ASCII code for that character.
4. A graph that plots both positive and negative values may be called ______________.
5. Data used in graphic display is often obtained from ___________.

Draw Graph

```
100  REM  ***--DRAW.GRAPH--***
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 150 TO ROW - 50 STEP 50
320  HPLOT C,I TO C - 4,I
330  NEXT I
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C + 250 STEP 25
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
460 :
700  REM  **--END--**
710  INPUT " ";NUL$
720  TEXT
730  END
```

Label Routine

```
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7: REM  WHITE
6020 INC = 50: REM  DISTANCE BETWEEN SCREEN POINTS
6030 LOW = 0:LOW = LOW + 25: REM  DISTANCE BETWEEN GRAPH POINTS
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
```

```
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C - 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM   ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 75 THEN 7010: REM  JUMP TO NEXT SUB-RTN
6120 LOW = LOW + 25: REM  INCREASE DATA VALUE
6130 INC = INC + 50: REM  INCREASE SCREEN POINT VALUE
6140  GOTO 6040: REM   GO BACK FOR MORE
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "SCATTER GRAPH"
7030  FOR T = 1 TO LEN (LABEL$)
7040 LB = ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8010 LC = C + 10
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040  FOR T = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0: REM  RESET SCREEN POSITION TO 0
8130 :
8140  RETURN
```

Scatter Graph (Maximum Value 30)

```
100  REM  ***--SCAT.30.GRAPH--***
110 :
```

```
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 150 TO ROW - 50 STEP 50
320  HPLOT C,I TO C - 4,1
330  NEXT I
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C + 250 STEP 25
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  **--PLOT DATA--**
610  HCOLOR= 3: REM  WHITE
620 RTIO = 5: REM  5 SCREEN POINTS = 1 DATA POINT
630  READ DTPT: REM  GET DATA VALUE
640  IF DTPT = 0 THEN 900: REM  GRAPH FINISHED
650 TPROW = ROW - (DTPT * RTIO): REM  TOP ROW = VALUE OF DATA
660  HPLOT C + 25 + INC,TPROW
670 INC = INC + 25: REM  INCREASE SCREEN POSITION FOR NEXT GRAPH POINT
680  GOTO 630: REM  GO BACK FOR MORE POINTS
690 :
695 :
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
```

```
1000  REM   **--DATA VALUES--**
1010  DATA  5,27,16,20,9,11,22,2,15,10,0
1020 :
1030 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7: REM  WHITE
6020 INC = 50: REM  DISTANCE BETWEEN SCREEN POINTS
6030 LOW = 0:LOW = LOW + 10: REM   DISTANCE BETWEEN GRAPH POINTS
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C - 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM   ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 30 THEN 7010: REM   JUMP TO NEXT SUB-RTN
6120 LOW = LOW + 10: REM   INCREASE DATA VALUE
6130 INC = INC + 50: REM   INCREASE SCREEN POINT VALUE
6140  GOTO 6040: REM   GO BACK FOR MORE
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "SCATTER GRAPH"
7030  FOR T = 1 TO  LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8010 LC = C + 10
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040  FOR T = 1 TO LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0: REM  RESET SCREEN POSITION TO 0
8130 :
8140  RETURN
```

Scatter Graph (Maximum Value 75)

```
100  REM  ***--SCAT.75.GRAPH--***
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
300  REM  **--SHORT HORIZ. LINES--**
310  FOR I = ROW - 150 TO ROW - 50 STEP 50
320  HPLOT C,I TO C - 4,I
330  NEXT I
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C + 250 STEP 25
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  **--PLOT DATA--**
610  HCOLOR= 3: REM  WHITE
620 RTIO = 2: REM  2 SCREEN POINTS = 1 DATA POINT
630  READ DTPT: REM  GET DATA VALUE
640  IF DTPT = 0 THEN 900: REM  GRAPH FINISHED
650 TPROW = ROW - (DTPT * RTIO): REM  TOP ROW = VALUE OF DATA
660  HPLOT C + 25 + INC,TPROW
670 INC = INC + 25: REM  INCREASE SCREEN POSITION FOR NEXT GRAPH POINT
680  GOTO 630: REM  GO BACK FOR MORE POINTS
690 :
695 :
```

```
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT   CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM  **--DATA VALUES--**
1010  DATA  5,27,16,20,9,11,22,2,15,10,0
1020 :
1030 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7: REM  WHITE
6020 INC = 50: REM  DISTANCE BETWEEN SCREEN POINTS
6030 LOW = 0:LOW = LOW + 25: REM  DISTANCE BETWEEN GRAPH POINTS
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
     $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
     ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C - 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM   ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 75 THEN 7010: REM  JUMP TO NEXT SUB-RTN
6120 LOW = LOW + 25: REM  INCREASE DATA VALUE
6130 INC = INC + 50: REM  INCREASE SCREEN POINT VALUE
6140  GOTO 6040: REM   GO BACK FOR MORE
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "SCATTER GRAPH"
7030  FOR T = 1 TO  LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8010 LC = C + 10
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040  FOR T = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
```

```
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0: REM  RESET SCREEN POSITION TO 0
8130 :
8140  RETURN
```

Line Graph (Maximum Value 30)

```
100  REM  ***--LINE.30.GRAPH--***
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 150 TO ROW - 50 STEP 50
320  HPLOT C,I TO C - 4,I
330  NEXT I
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C + 250 STEP 25
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  **--PLOT DATA--**
610  HCOLOR= 3: REM  WHITE
620 RTIO = 5: REM  5 SCREEN POINTS = 1 DATA POINT
630  READ DTPT: REM  GET DATA VALUE
```

```
640  IF DTPT = 0 THEN 900: REM  GRAPH FINISHED
650 TPROW = ROW - (DTPT * RTIO): REM  TOP ROW = VALUE OF DATA
655  IF INC = 0 THEN  HPLOT C,TPROW: GOTO 670
660  HPLOT  TO C + INC,TPROW
670 INC = INC + 25: REM  INCREASE SCREEN POSITION FOR NEXT GRAPH POINT
680  GOTO 630: REM  GO BACK FOR MORE POINTS
690 :
695 :
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM  **--DATA VALUES--**
1010  DATA 5,27,16,20,9,11,22,2,15,10,0
1020 :
1030 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7: REM  WHITE
6020 INC = 50: REM  DISTANCE BETWEEN SCREEN POINTS
6030 LOW = 0:LOW = LOW + 10: REM   DISTANCE BETWEEN GRAPH POINTS
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C - 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM  ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 30 THEN 7010: REM  JUMP TO NEXT SUB-RTN
6120 LOW = LOW + 10: REM  INCREASE DATA VALUE
6130 INC = INC + 50: REM  INCREASE SCREEN POINT VALUE
6140  GOTO 6040: REM   GO BACK FOR MORE
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "LINE GRAPH"
7030  FOR T = 1 TO  LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
```

```
8010 LC = C - 8
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040  FOR T = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0: REM  RESET SCREEN POSITION TO 0
8130 :
8140  RETURN
```

Line Graph (Maximum Value 75)

```
100  REM  ***--LINE.75.GRAPH--***
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 150 TO ROW - 50 STEP 50
320  HPLOT C,I TO C - 4,I
330  NEXT I
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C + 250 STEP 25
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
```

```
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  **--PLOT DATA--**
610  HCOLOR= 3: REM   WHITE
620 RTIO = 2: REM   2 SCREEN POINTS = 1 DATA POINT
630  READ DTPT: REM  GET DATA VALUE
640  IF DTPT = 0 THEN 900: REM  GRAPH FINISHED
650 TPROW = ROW - (DTPT * RTIO): REM  TOP ROW = VALUE OF DATA
655  IF INC = 0 THEN  HPLOT C,TPROW: GOTO 670
660  HPLOT  TO C + INC,TPROW
670 INC = INC + 25: REM  INCREASE SCREEN POSITION FOR NEXT GRAPH POINT
680  GOTO 630: REM  GO BACK FOR MORE POINTS
690 :
695 :
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM   **--DATA VALUES--**
1010  DATA  5,27,16,20,9,11,22,2,15,10,0
1020 :
1030 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7: REM  WHITE
6020 INC = 50: REM  DISTANCE BETWEEN SCREEN POINTS
6030 LOW = 0:LOW = LOW + 25: REM    DISTANCE BETWEEN GRAPH POINTS
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C - 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM   ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 75 THEN 7010: REM    JUMP TO NEXT SUB-RTN
6120 LOW = LOW + 25: REM    INCREASE DATA VALUE
6130 INC = INC + 50: REM   INCREASE SCREEN POINT VALUE
6140  GOTO 6040: REM   GO BACK FOR MORE
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
```

```
7020 LABEL$ = "LINE GRAPH"
7030  FOR T = 1 TO  LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8010 LC = C - 8
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040  FOR T = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0: REM  RESET SCREEN POSITION TO 0
8130 :
8140  RETURN
```

Deviation Graph

```
100  REM   ***--DEVIT.15.GRAPH--***
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
225  HCOLOR= 7: HPLOT C,ROW - 75 TO C + 250,ROW - 75
230 :
240 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 150 TO ROW - 25 STEP 25
```

```
320  HPLOT C,I TO C + 4,I
330  NEXT I
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C + 250 STEP 25
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  **--PLOT DATA--**
610 INC = 25
620 RTIO = 5: REM  5 SCREEN POINTS = 1 DATA POINT
630  READ DTPT: REM  GET DATA VALUE
640  IF DTPT = 0 THEN 900: REM  GRAPH FINISHED
650 TPROW = (ROW - 75) - (DTPT * RTIO): REM   TOP ROW = VALUE OF DATA
655  IF DTPT > 0 THEN  HCOLOR= 6
657  IF DTPT < 0 THEN  HCOLOR= 5
660  HPLOT C + INC,ROW - 75 TO C + INC,TPROW
665  HPLOT C + INC + 1,ROW - 75 TO C + INC + 1,TPROW
666  HPLOT C + INC - 1,ROW - 75 TO C + INC - 1,TPROW
670 INC = INC + 25: REM  INCREASE SCREEN POSITION FOR NEXT GRAPH POINT
680  GOTO 630: REM  GO BACK FOR MORE POINTS
690 :
695 :
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM   **--DATA VALUES--**
1010  DATA   5,12,-9,-5,9,11,-7,2,-13,10,0
1020 :
1030 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7: REM  WHITE
6020 INC = 0: REM   DISTANCE BETWEEN SCREEN POINTS
6030 LOW = 0:LOW = LOW - 15: REM    DISTANCE BETWEEN GRAPH POINTS
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
```

```
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
   ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C - 7,ROW - INC - 3
6090  DRAW D2 AT C - 14,ROW - INC - 3
6100  DRAW D3 AT C - 21,ROW - INC - 3
6110  IF LOW = 15 THEN 7010: REM    JUMP TO NEXT SUB-RTN
6120 LOW = LOW + 5: REM    INCREASE DATA VALUE
6130 INC = INC + 25: REM    INCREASE SCREEN POINT VALUE
6140  GOTO 6040: REM   GO BACK FOR MORE
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "DEVIATION GRAPH"
7030  FOR T = 1 TO  LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8010 LC = C + 10
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040  FOR T = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0: REM  RESET SCREEN POSITION TO 0
8130 :
8140  RETURN
```

Column Graph (Maximum Value 30)

```
100  REM   ***--COL.30.GRAPH--***
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
```

```
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 150 TO ROW - 50 STEP 50
320  HPLOT C,I TO C - 4,I
330  NEXT I
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C = 250 STEP 25
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  *--PLOT DATA--*
610 WD = 10: REM   WIDTH OF COLUMNS
620 RTIO = 5: REM  EACH DATA POINT = 5 SCREEN POINTS
630 ALT$ = "BLUE": HCOLOR= 6
640  READ DTPT: REM   GET DATA VALUE
650  IF DTPT = 0 THEN 900: REM    END OF DATA
660 TPROW = ROW - (DTPT * RTIO): REM   TOP ROW INDICATING VALUE OF DATA
670  HPLOT C + 15 + INC,ROW TO C + 15 + INC,TPROW TO C + 15 + INC + WD,T
    PROW TO C + 15 + INC + WD,ROW
680 :
690  REM   *--FILL IN COLUMNS--*
700  FOR Z = C + 15 + INC TO C + 15 + INC + WD
710  HPLOT Z,ROW TO Z,TPROW
720  NEXT Z
730 :
740 INC = INC + 25: REM  MOVE TO NEXT POSITION
750  IF ALT$ = "ORANGE" THEN  HCOLOR= 6:ALT$ = "BLUE": GOTO 770
760  IF ALT$ = "BLUE" THEN  HCOLOR= 5:ALT$ = "ORANGE"
770  GOTO 640: REM  GO BACK FOR MORE INFO
780 :
790 :
```

```
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM   **--DATA VALUES--**
1010  DATA  5,27,16,20,9,11,22,2,15,10,0
1020 :
1030 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7: REM  WHITE
6020 INC = 50: REM  DISTANCE BETWEEN SCREEN POINTS
6030 LOW = 0:LOW = LOW + 10: REM   DISTANCE BETWEEN GRAPH POINTS
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C - 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM   ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 30 THEN 7010: REM   JUMP TO NEXT SUB-RTN
6120 LOW = LOW + 10: REM   INCREASE DATA VALUE
6130 INC = INC + 50: REM   INCREASE SCREEN POINT VALUE
6140  GOTO 6040: REM   GO BACK FOR MORE
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "COLUMN GRAPH"
7030  FOR T = 1 TO LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8010 LC = C + 10
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040  FOR 5 = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
```

```
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0: REM  RESET SCREEN POSITION TO 0
8130 :
8140  RETURN
```

Column Graph (Maximum Value 75)

```
100  REM   ***--COL.75.GRAPH--***
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 150 TO ROW - 50 STEP 50
320  HPLOT C,I TO C - 4,I
330  NEXT I
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C + 250 STEP 25
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  *--PLOT DATA--*
610 WD = 10: REM   WIDTH OF COLUMNS
620 RTIO = 2: REM   EACH DATA POINT = 2 SCREEN POINTS
```

```
630 ALT$ = "BLUE": HCOLOR= 6
640  READ DTPT: REM   GET DATA VALUE
650  IF DTPT = 0 THEN 900: REM    END OF DATA
660 TPROW = ROW - (DTPT * RTIO): REM   TOP ROW INDICATING VALUE OF DATA
670  HPLOT C + 15 + INC,ROW TO C + 15 + INC,TPROW TO C + 15 + INC + WD,T
    PROW TO C + 15 + INC + WD,ROW
680 :
690  REM   *--FILL IN COLUMNS--*
700  FOR Z = C + 15 + INC TO C + 15 + INC + WD
710  HPLOT Z,ROW TO Z,TPROW
720  NEXT Z
730 :
740 INC = INC + 25: REM  MOVE TO NEXT POSITION
750  IF ALT$ = "ORANGE" THEN  HCOLOR= 6:ALT$ = "BLUE": GOTO 770
760  IF ALT$ = "BLUE" THEN  HCOLOR= 5:ALT$ = "ORANGE"
770  GOTO 640: REM  GO BACK FOR MORE INFO
780 :
790 :
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM  **--DATA VALUES--**
1010  DATA  5,27,16,20,9,11,22,2,15,10,0
1020 :
1030 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7: REM  WHITE
6020 INC = 50: REM  DISTANCE BETWEEN SCREEN POINTS
6030 LOW = 0:LOW = LOW + 25: REM  DISTANCE BETWEEN GRAPH POINTS
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C - 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM   ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 75 THEN 7010: REM  JUMP TO NEXT SUB-RTN
6120 LOW = LOW + 25: REM  INCREASE DATA VALUE
6130 INC = INC + 50: REM  INCREASE SCREEN POINT VALUE
6140  GOTO 6040: REM   GO BACK FOR MORE
6150 :
6160 :
7000  REM  **--TOP LABEL--**
```

```
7010 LC = 95:LR = 5
7020 LABEL$ = "COLUMN GRAPH"
7030  FOR T = 1 TO LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8010 LC = C + 10
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040  FOR T = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0: REM  RESET SCREEN POSITION TO 0
8130 :
8140  RETURN
```

Area Graph (Maximum Value 30)

```
100  REM   ***--AREA.30.GRAPH--***
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW - 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 150 TO ROW - 50 STEP 50
320  HPLOT C,I TO C - 4,I
```

```
330  NEXT I
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C + 250 STEP 25
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  *--PLOT DATA--*
605 RTIO = 5: REM   EACH DATA POINT = 5 SCREEN POINTS
610  HCOLOR= 5
615  READ DTPT: REM   GET DATA VALUE
620  IF DTPT = 0 THEN 750: REM    END OF FIRST DATA
625 TPROW = ROW - (DTPT * RTIO): REM   TOP ROW INDICATING VALUE OF DATA
630  IF INC = 0 THEN  HPLOT C,TPROW: GOTO 730
635  HPLOT  TO C + INC,TPROW
640 :
645  REM  *--TEST FOR CURRENT POINT <> PREVIOUS POINT--*
650  IF TPROW > OT THEN L = TP - OT: GOTO 700
655  IF TPROW < OT THEN L = OT - TPROW
660  IF TPROW = OT THEN L = TPROW
665 :
670  REM   *--FILL LEFT SIDE--*
675  FOR Q = 0 TO L: HPLOT C + (INC - 25),OT TO C + INC,TPROW + Q: NEXT
   Q
680 D = ROW - OT
685  FOR W = 0 TO D: HPLOT C + (INC - 25),OT + W TO C + INC,OT + W: NEXT
   W
690  GOTO 725
695 :
700  REM   *--FILL RIGHT SIDE--*
705  FOR Q = 0 TO L: HPLOT C + INC,TPROW TO C + (INC - 25),OT + Q: NEXT
   Q
710 D = ROW - TPROW
715  FOR W = 0 TO D: HPLOT C + (INC - 25),TP + W TO C + INC,TP + W: NEXT
   W
720 :
725  HPLOT C + INC,TPROW
730 INC = INC + 25: REM   MOVE TO NEXT POSITION
735 OT = TPROW: REM  PREVIOUS POINT
740  GOTO 615: REM   GO BACK FOR MORE INFO
```

```
745 :
750  REM   *--SECOND LINE--*
755 INC = 0
760  HCOLOR= 6
765  READ DTPT: REM    GET DATA VALUE
770  IF DTPT = 0 THEN 900: REM    END OF SECOND DATA
775 TPROW = ROW - (DTPT * RTIO): REM    TOP ROW INDICATING VALUE OF DATA
780  IF INC = 0 THEN  HPLOT C,TPROW: GOTO 795
785  HPLOT  TO C + INC,TPROW
790 :
795 INC = INC + 25: REM    MOVE TO NEXT POSITION
800  GOTO 765: REM   GO BACK FOR MORE INFO
805 :
810 :
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM  **--DATA--**
1010  DATA     5,27,16,20,9,11,22,2,15,10,0
1020  DATA    20,29,25,22,25,15,24,20,24,12,0
1030 :
1040 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7
6020 INC = 50
6030 LOW = 0:LOW = LOW + 10
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C = 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM  ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 30 THEN 7010
6120 LOW = LOW + 10
6130 INC = INC + 50
6140  GOTO 6040
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "AREA GRAPH"
```

```
7030  FOR T = 1 TO  LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8010 LC = C - 10
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040  FOR T = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0
8130 :
8140  RETURN
```

Area Graph (Maximum Value 75)

```
100  REM    ***--AREA.75.GRAPH--***
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 150 TO ROW - 50 STEP 50
320  HPLOT C,I TO C - 4,I
330  NEXT I
```

```
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C + 250 STEP 25
430  HPLOT I,ROW - 4, TO I,ROW
440  NEXT I
450 :
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  *--PLOT DATA--*
605 RTIO = 2: REM    EACH DATA POINT = 2 SCREEN POINTS
610  HCOLOR= 3
615  READ DTPT: REM   GET DATA VALUE
620  IF DTPT = 0 THEN 750: REM    END OF FIRST DATA
625 TPROW = ROW - (DTPT * RTIO): REM   TOP ROW INDICATING VALUE OF DATA
630  IF INC = 0 THEN  HPLOT C,TPROW: GOTO 730
635  HPLOT  TO C + INC,TPROW
640 :
645  REM  *--TEST FOR CURRENT POINT <> PREVIOUS POINT--*
650  IF TPROW > OT THEN L = TP - OT: GOTO 700
655  IF TPROW < OT THEN L = OT - TPROW
660  IF TPROW = OT THEN L = TPROW
665 :
670  REM   *--FILL LEFT SIDE--*
675  FOR Q = 0 TO L: HPLOT C + (INC - 25),OT TO C + INC,TPROW + Q: NEXT
    Q
680 D = ROW - OT
685  FOR W = 0 TO D: HPLOT C + (INC - 25),0T + W TO C + INC,OT + W: NEXT
    W
690  GOTO 725
695 :
700  REM   *--FILL RIGHT SIDE--*
705  FOR Q = 0 TO L: HPLOT C + INC,TPROW TO C + (INC - 25),OT + Q: NEXT
    Q
710 D = ROW - TPROW
715  FOR W = 0 TO D: HPLOT C + (INC - 25),TP + W TO C + INC,TP + W: NEXT
    W
720 :
725  HPLOT C + INC,TPROW
730 INC = INC + 25: REM   MOVE TO NEXT POSITION
735 OT = TPROW: REM  PREVIOUS POINT
740  GOTO 615: REM    GO BACK FOR MORE INFO
745 :
750  REM   *--SECOND LINE--*
```

```
755 INC = 0
760  HCOLOR= 6
765  READ DTPT: REM    GET DATA VALUE
770  IF DTPT = 0 THEN 900: REM    END OF SECOND DATA
775 TPROW = ROW - (DTPT * RTIO): REM    TOP ROW INDICATING VALUE OF DATA
780  IF INC = 0 THEN  HPLOT C,TPROW: GOTO 795
785  HPLOT  TO C + INC,TPROW
790 :
795 INC = INC + 25: REM    MOVE TO NEXT POSITION
800  GOTO 765: REM   GO BACK FOR MORE INFO
805 :
810 :
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM  **--DATA--**
1010  DATA     5,27,16,20,9,11,22,2,15,10,0
1020  DATA    20,29,25,22,25,15,24,20,24,12,0
1030 :
1040 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7
6020 INC = 50
6030 LOW = 0:LOW = LOW + 25
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C - 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM  ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 75 THEN 7010
6120 LOW - LOW + 25
6130 INC = INC + 50
6140  GOTO 6040
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "AREA GRAPH"
7030  FOR T = 1 TO  LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
```

```
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8010 LC = C - 10
8020 LR = ROW + 10
8030 LABEL$ = "SEPOCTNOVDECJANFEBMARAPRMAYJUN"
8040  FOR T = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0
8130 :
8140  RETURN
```

Bar Graph
(Maximum Value 30)

```
100  REM   ***--BAR.30.GRAPH--***
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 157: REM  ROW VALUE
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
250  GOTO 400: REM  SKIP FOR BAR GRAPHS
260 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 150 TO ROW - 50 STEP 50
320  HPLOT C,I TO C - 4,I
330  NEXT I
```

```
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C TO C + 240 STEP 80
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  *--PLOT DATA--*
605 WD = 10: REM  WIDTH OF BARS
610 RTIO = 8: REM   EACH DATA POINT = 8 SCREEN POINTS
615 KOLR = 1
620  READ DTPT: REM   GET DATA VALUE
625  IF KOLR = 4 THEN KOLR = 5
630  HCOLOR= KOLR
635  IF DTPT = 0 THEN 900: REM    END OF DATA
640 TPCLM = C + (DTPT * RTIO): REM    TOP COLUMN INDICATING VALUE OF DAT
    A
645  HPLOT C,ROW - 15 - INC TO TPCLM,ROW - 15 - INC TO TPCLM,ROW - 15 -
    INC - WD TO C,ROW - 15 - INC - WD
650 :
655  REM    *--FILL IN BARS--*
660  FOR Z = ROW - 15 - INC TO ROW - 15 - INC - WD STEP  - 1
665  HPLOT C,Z TO TPCLM,Z
670  NEXT Z
675 :
680 INC = INC + 25: REM    MOVE TO NEXT POSITION
685 KOLR = KOLR + 1
690  GOTO 620: REM  GO BACK FOR MORE INFO
695 :
700 :
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM  **--DATA--**
1010  DATA    5,27,16,20,9,0,11,22,2,15,10,0
1020 :
1030 :
6000  REM   **--BOTTOM LABEL--**
6010  HCOLOR= 7
```

```
6020 INC = 80
6030 LOW = 0:LOW = LOW + 10
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1,)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C + INC + 3,ROW + 5
6090  DRAW D2 AT C + INC + 3 - 9,ROW + 5
6100  REM  ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 30 THEN 7010
6120 LOW = LOW + 10
6130 INC = INC + 80
6140  GOTO 6040
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "BAR GRAPH"
7030  FOR T = 1 TO LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM   **--LEFT SIDE LABEL--**
8010 LC = 0
8020 LR = ROW - 25 + 2
8030 LABEL$ = "SEPOCTNOVDECJAN"
8040  FOR T = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
8090 LR = LR - 25
8100  NEXT T
8110 :
8120 INC = 0
8130 :
8140  RETURN
```

Hi/Low/Close Graph

```
100  REM   ***--HI.LW.CLS.GRAPH--***
```

```
110 :
120 :
130  REM  **--SET UP GRAPHICS--**
140  HGR
150  POKE  - 16302,0: REM  FULL SCREEN GRAPHICS
160  HCOLOR= 5: REM  BROWN
170 C = 25: REM  COLUMN VALUE
180 ROW = 151: REM   ROW VALUE
185 X = C + 25:LOW = 300
190 :
195 :
200  REM  **--DRAW GRAPH LINES--**
210  HPLOT C,ROW - 150 TO C,ROW: REM  LEFT VERTICAL LINE
220  HPLOT C,ROW TO C + 250,ROW: REM  BOTTOM HORIZ.LINE
230 :
240 :
300  REM  **--SHORT HORIZ.LINES--**
310  FOR I = ROW - 144 TO ROW - 7 STEP 8
320  HPLOT C,I TO C - 4,I
330  NEXT I
340 :
350 :
400  REM  **--SHORT VERT.LINES--**
410  HCOLOR= 7
420  FOR I = C + 25 TO C + 250 STEP 25
430  HPLOT I,ROW - 4 TO I,ROW
440  NEXT I
450 :
460 :
500  REM  **--LABEL ROUTINE--**
510  GOSUB 6000: REM  LABEL ROUTINES
520 :
530 :
600  REM  **--PLOT HI/LOW/CLOSE PRICES--**
605 ALT$ = "BLUE"
610  HCOLOR= 7
615  FOR K = 1 TO 10
620  READ DAY,DHIGH,DLOW,DCLOSE
625 DAY = X
630 Q = DLOW: GOSUB 4000:DLOW = PT
635  IF DLOW > 191 THEN DLOW = 191
640  IF DLOW < 0 THEN DLOW = 0
645  HPLOT DAY,DLOW
650 Q = DHIGH: GOSUB 4000:DHIGH = PT
655  IF DHIGH < 0 THEN DHIGH = 0
660  IF DHIGH > 191 THEN DHIGH = 191
665  HPLOT DAY,DHIGH
670 Q = DCLOSE: GOSUB 4000:DCLOSE = PT
```

```
675  IF DCLOSE < 0 THEN DCLOSE = 1
680  IF DCLOSE > 191 THEN DCLOSE = 190
685  HPLOT DAY,DCLOSE
690  HPLOT DAY,DHIGH TO DAY,DLOW: HPLOT DAY - 1,DHIGH TO DAY - 1,DLOW: HPLOT
    DAY + 1,DHIGH TO DAY + 1,DLOW
705  HPLOT DAY - 4,DCLOSE TO DAY + 4,DCLOSE
710  HCOLOR= 7
715 X = X + 25
720  NEXT K
725 :
730 :
900  REM  **--END--**
910  INPUT " ";NUL$
920  TEXT
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM  **--DATA--**
1010  DATA  1,397,393,395
1020  DATA  2,405,387,390
1030  DATA  3,427,390,423
1040  DATA  4,454,414,415
1050  DATA  5,427,415,423
1060  DATA  6,450,425,443
1070  DATA  7,472,444,470
1080  DATA  8,482,461,471
1090  DATA  9,485,466,480
1100  DATA  10,510,481,490
1110 :
1120 :
4000  REM  **--DETERMINE POINTS--**
4010 R =  INT ((Q - LOW) / 10)
4020 R1 = (Q - LOW) - (R * 10)
4030 P1 = (R * 8) + R1
4040 PT = 183 - P1
4050  RETURN
4060 :
4070 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7
6020 INC = 8
6030 LOW = 0:LOW = LOW + 35
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
```

```
6080  DRAW D1 AT C - 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM  ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 52 THEN 7000
6120 LOW = LOW + 1
6130 INC = INC + 8
6140  GOTO 6040
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7010 LC = 95:LR = 5
7020 LABEL$ = "HI/LOW/CLOSE GRAPH"
7030  FOR T = 1 TO  LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8010 LC = C + 11
8020 LR = ROW + 10
8030 LABEL$ = "MONTUEWEDTHRFRIMONTUEWEDTHRFRI"
8040  FOR T = 1 TO  LEN (LABEL$) STEP 3
8050 L1 =  ASC ( MID$ (LABEL$,T,1))
8060 L2 =  ASC ( MID$ (LABEL$,T + 1,1))
8070 L3 =  ASC ( MID$ (LABEL$,T + 2,1))
8080  DRAW L1 AT LC,LR: DRAW L2 AT LC + 7,LR: DRAW L3 AT LC + 14,LR
8090 LC = LC + 25
8100  NEXT T
8110 :
8120 INC = 0:LOW = 300
8130 :
8140  RETURN
```

Pie Graph

```
100  REM   ***--PIE.GRAPH--***
110 :
120 :
130  REM  **--INITIAL VALUES--**
140 PI = 3.1415926:::: REM  VALUE FOR PI
150 COLUMN1 = 141:::: REM  FIRST COLUMN POSITION
160 ROW1 = 80:::::::: REM  FIRST ROW POSITION
170 RADIUS = 70:::::: REM  SIZE OF CIRCLE
180 NUMBER = 512::::: REM  NUMBER OF POINTS ON CIRCLE
```

```
190 CC = 2 * PI::::::: REM  EXTENT OF LOOP
200 INC = CC / NUMBER: REM  INCREMENT IN LOOP
205 Y =  - 75
210 :
220 :
500  REM  **--SET UP GRAPHICS--**
510  HGR : POKE  - 16302,0: REM  FULL SCREEN
530 :
540 :
600  REM  **--GET DATA--**
610  FOR I = 1 TO 5
620  READ SEC(I)
630 TA = TA + SEC(I)
640  NEXT I
650 :
660 :
670  REM   **--FILL CIRCLE--**
680 Z = 1
690 EN = (SEC(Z) / TA) * CC
700  READ KOLR: HCOLOR= KOLR
710  FOR I = 0 TO CC STEP INC
720  IF I > EN THEN  GOSUB 5000: REM  NEW SECTION
730 CLMN = RADIUS *  SIN (I)
740 ::RW = RADIUS *  COS (I)
750 RW = RW * .83: REM   ADJUST FOR ELLIPSE EFFECT
760  HPLOT COLUMN1,ROW1 TO COLUMN1 + CLMN,ROW1 + RW
770  NEXT I
780 :
790 :
800 Z = 5: GOSUB 5000: REM  LAST SECTION
810  GOSUB 7000: REM  TOP LABEL
820 :
830 :
900  REM  **--END--**
910  INPUT " ";L$
920  TEXT : HOME
930  PRINT  CHR$ (4);"RUN GRAPH.MENU"
940 :
950 :
1000  REM  **--SECTIONS OF PIE CHART--**
1010  DATA   27,12,60,47,20
1020 :
1030 :
2000  REM  **--COLOR FOR DIFF.PIE SECTIONS--**
2010  DATA      1,3,5,6,2,0
2020 :
2030 :
5000  REM  **--NEXT SECTION OF PIE CHART--**
```

```
5010 EN = ((SEC(Z) / TA) * CC) + I: REM  ENDING VALUE
5020  READ KOLR: HCOLOR= KOLR
5030 :
5040  REM  *--LABEL SECTION--*
5050  IF Z = 1 THEN LABEL$ = "CAR":LC = COLUMN1 + CLMN - 10:LR = ROW1 +
    RW + 25
5060  IF Z = 2 THEN LABEL$ = "FOOD":LC = COLUMN1 + CLMN + 12:LR = ROW1 +
    RW + 18
5070  IF Z = 3 THEN LABEL$ = "CHURCH":LC = COLUMN1 + CLMN + 15:LR = ROW1
     + RW + 5
5080  IF Z = 4 THEN LABEL$ = "HOUSE":LN =  LEN (LABEL$):LC = (COLUMN1 +
    CLMN) - 3 - (LN * 8):LR = (ROW1 + RW) - 40
5090  IF Z = 5 THEN LABEL$ = "UTILITIES":LN =  LEN (LABEL$):LC = COLUMN1
     - (LN * 15):LR = (ROW1 + RW) - 5
5100  HCOLOR= 3
5110 Z = Z + 1
5120  GOTO 8000: REM  SKIP LEFT & TOP LABELS
5130 :
5140 :
6000  REM  **--LEFT SIDE LABEL--**
6010  HCOLOR= 7
6020 INC = 50
6030 LOW = 0:LOW = LOW + 10
6040 LOW$ =  STR$ (LOW):LN =  LEN (LOW$)
6050  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
6060  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
6070  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC (MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
6080  DRAW D1 AT C - 14,ROW - INC - 3
6090  DRAW D2 AT C - 21,ROW - INC - 3
6100  REM  ETC. DRAW D3 AT C - 28,ROW - INC
6110  IF LOW = 30 THEN 7010
6120 LOW = LOW + 10
6130 INC = INC + 50
6140  GOTO 6040
6150 :
6160 :
7000  REM  **--TOP LABEL--**
7005  HCOLOR= 3
7010 LC = 65:LR = 5
7020 LABEL$ = "PIE OR CIRCLE CHART"
7030  FOR T = 1 TO  LEN (LABEL$)
7040 LB =  ASC ( MID$ (LABEL$,T,1))
7050  DRAW LB AT LC,LR
7060 LC = LC + 9
7070  NEXT T
7072 INC = 0
```

```
7075  RETURN
7080 :
7090 :
8000  REM  **--BOTTOM LABEL--**
8005 LN =  LEN (LABEL$)
8007  REM IF I > CC / 2 THEN LC = (COLUMN1 + CLMN) - 3 - (LN * 9):LR = (
    ROW1 + RW) - 10
8010  REM LC = C + 10
8020  REM LR = ROW + 10
8030  FOR T = 1 TO LEN (LABEL$)
8040 LB =  ASC ( MID$ (LABEL$,T,1))
8050  DRAW LB AT LC,LR
8060 LC = LC + 9
8070  NEXT T
8072  HCOLOR= KOLR
8075  RETURN
```

Grades Graph

```
100  REM   ***--GRADES.GRAPH--***
110 :
120 :
130  REM  **--INITIALIZATION--**
140  DIM Z(150),A(150)
150  HOME : VTAB 10
160 C$ = "4":SUB$ = "ENGLISH"
170 MAX = 313:D$ =  CHR$ (4)
180 :
190 :
200  REM  **--GET DATA--**
210 I = 1
220  READ Z(I)
230  IF Z(I) = 99999 THEN 500
240 PER = Z(I) / MAX
250 Z(I) =  INT (PER * 100 + .5)
260 PZ = PZ + Z(I)
270 M = M + (Z(I)) ^ 2
280  PRINT ".";:I = I + 1
290  GOTO 220
300 :
310 :
500  REM  **--DO CALCULATIONS--**
510 N = I - 1
520  DIM L(N),R(N)
530 AVE = PZ / N
540 R = PZ / N
```

```
550 V = (M - (PZ) ^ 2 / N) / N
560 R =  INT (R):SD =  SQR (V)
570 :
580 :
600  REM  **--SORT SCORES--**
610  GOSUB 5000: REM  SORT
620 :
630 :
1000  REM  **--HIRES GRAPH OF SCORES--**
1010 :
1020  REM  PRINT CHR$(4);"BLOAD ASCII.SHP.TABLE"
1030  POKE 232,0: POKE 233,96: REM SHAPE TABLE ADDRESS
1040  HGR
1050  POKE  - 16302,0: REM  FULL SCREEN
1070  HCOLOR= 5
1080  SCALE= 1
1090  ROT= 0
1100 :
1110 :
1500  REM  **--DRAW VERTICAL DIST.LINE--**
1510  HPLOT 51,10 TO 51,180
1520  FOR I = 10 TO 170 STEP 20
1530  HPLOT 47,I TO 55,I
1540  NEXT I
1550 :
1560 :
1570  REM  **--LABEL DIST.LINE--**
1580 Y = 167
1590  HCOLOR= 7
1600  FOR I = 2 TO 9
1610 I$ =  STR$ (I)
1620  DRAW  ASC (I$) AT 25,Y: DRAW 48 AT 32,Y: DRAW 37 AT 39,Y
1630 Y = Y - 20
1640  NEXT I
1650  DRAW 49 AT 18,7: DRAW 48 AT 25,7: DRAW 48 AT 32,7: DRAW 37 AT 39,7

1660 :
1670 :
2000  REM  **--LABEL GRAPH--**
2010 X = 110:Y = 10
2020 LB$ = "PERIOD " + C$ + " " + SUB$
2030  IF LB$ = "END" THEN 2500
2040  FOR I = 1 TO LEN (LB$)
2050 LB =  ASC ( MID$ (LB$,I,1))
2060  IF LB = 32 AND Y > 20 THEN Y = Y + 20:X = X - 100
2070  DRAW LB AT X,Y
2080 X = X + 10
2090  NEXT I
```

```
2100  IF Y < 20 THEN LB$ = "PERCENT DISTRIBUTION"
2110  IF Y > 20 THEN LB$ = "END"
2120 Y = Y + 25:X = 150: GOTO 2030
2130 :
2140 :
2500  REM  **--DISPLAY CALCULATION RESULTS--**
2510 X = 110:Y = 150
2520 LB$ = "MEAN = " +  STR$ (R) + "*"
2530  IF LB$ = "END" THEN 3000
2540  FOR I = 1 TO  LEN (LB$)
2550 LB =  ASC ( MID$ (LB$,I,1))
2560  IF LB = 42 THEN 2590
2570  DRAW LB AT X,Y
2580 X = X + 10
2590  NEXT I
2600  IF Y = 150 THEN LB$ = "VARIANCE = " + STR$ ( INT (V)) + "*"
2610  IF Y = 160 THEN LB$ = "SD =" + LEFT$ ( STR$ (SD),5) + "*"
2620  IF Y = 170 THEN LB$ = "END"
2630 Y = Y + 10:X = 110: GOTO 2530
2640 :
2650 :
3000  REM  **--PLOT DISTRIBUTION--**
3010 X = 60
3020  FOR K = 1 TO N
3030  IF A(K) <  > ZZ THEN X = 60
3040 Y = 180 - ((A(K) - 15) * 2)
3050  IF Y > 180 THEN Y = 180
3060  HPLOT X,Y TO X + 3,Y
3070 X = X + 10
3080 ZZ = A(K)
3090  NEXT K
3110 :
3120 :
3130 :
4000  REM  **--END--**
4010  INPUT " ";L$
4020  TEXT : HOME
4030  PRINT  CHR$ (4);"RUN GRAPH.MENU"
4040 :
4050 :
4060 :
5000  REM  **--SORT ROUTINE--**
5010  FOR K = 1 TO N
5020 A(K) = Z(K)
5030  NEXT K
5040 S1 = 1
5050  INVERSE
5060  HTAB 5: PRINT "WORKING--PLEASE DON'T TOUCH!!"
```

```
5070  NORMAL
5080 L(1) = 1
5090 R(1) = N
5100 L1 = L(S1)
5110 R1 = R(S1)
5120 S1 = S1 - 1
5130 L2 = L1
5140 R2 = R1
5150 X = A( INT ((L1 + R1) / 2))
5160 C = C + 1
5170  IF A(L2) = X OR A(L2) > X THEN 5200
5180 L2 = L2 + 1
5190  GOTO 5160
5200 C = C1
5210  IF X = A(R2) OR X > A(R2) THEN 5240
5220 R2 = R2 - 1
5230  GOTO 5200
5240  IF L2 > R2 THEN 5310
5250 S = S + 1
5260 T = A(L2)
5270 A(L2) = A(R2)
5280 A(R2) = T
5290 L2 = L2 + 1
5300 R2 = R2 - 1
5310  IF L2 = R2 OR L2 < R2 THEN 5160
5320  IF L2 = R1 OR L2 > R1 THEN 5360
5330 S1 = S1 + 1
5340 L(S1) = L2
5350 R(S1) = R1
5360 R1 = R2
5370  IF L1 < R1 THEN 5130
5380  IF S1 > 0 THEN 5100
5390  REM   SORT COMPLETE
5400  RETURN
5410 :
5420 :
9000  REM  **--DATA VALUES--**
9010  DATA  250,197,267,225,242,216,170,294,185,225
9020  DATA  255,257,272,213,254,266,233,280,286,192
9030  DATA  217,273,259,244,251,219,267,313,99999
```

Graph Menu

```
100  REM   ***--GRAPH.MENU--***
110 :
120 :
```

```
130 TB = 2: DIM A(20)
140 D$ =  CHR$ (4): REM  CONTROL D
150 PK =  PEEK (800)
152  IF PK = 244 THEN 180
154  POKE 800,244
156  PRINT D$;"BLOAD ASCII.SHP.TABLE"
158  POKE 232,0: POKE 233,96: REM  SHP TBL ADDRESS
160  SCALE= 1: ROT= 0: HCOLOR= 3
165 :
170 :
180  REM   **--GRAPH EXAMPLE SYSTEM MENU--**
190  HOME
200  FOR I = 2 TO 79: PRINT "_";: NEXT I
210  PRINT
220  VTAB 3
230  HTAB TB
240  PRINT "GRAPH EXAMPLES";
250 TITLE$ = "GRAPH EXAMPLE SYSTEM MENU"
260 LN =  LEN (TITLE$)
270 CENTER = (80 - LN) / 2: REM  CENTER TITLE
280  HTAB CENTER
290  INVERSE
300  PRINT TITLE$;
310  NORMAL
320  PRINT  SPC( 10): REM   10 SPACES
330  PRINT "Select Action"
340  PRINT
345  PRINT : HTAB TB; PRINT "1.  Line Graph Example
          6.  Hi/Low/Cls Graph Example"
350  PRINT : HTAB TB: PRINT "2.  Column Graph Example
          7.  Area Graph Example"
360  PRINT : HTAB TB: PRINT "3.  Bar Graph Example
          8.  Pie Graph Example"
370  PRINT : HTAB TB: PRINT "4.  Scatter Graph Example
          9.  Grades Graph Example"
380  PRINT : HTAB TB: PRINT "5.  Deviation Graph Example"
390  PRINT : PRINT
410  VTAB 18: HTAB TB: PRINT "Type your Selection ";: INVERSE : PRINT "(
    1-9)";: NORMAL : PRINT " and press ";: INVERSE : PRINT "RETURN";: NORMAL
    : PRINT ":";
420  NORMAL : PRINT " ---> ";
430 VERT =  PEEK (37):HRIZ =  PEEK (1403): REM  $37 CONTAINS VERTICAL CU
    RSOR POSITION; $1403 CONTAINS HORIZONTAL CURSOR POSITION FOR 80 COL.
     //e--$36 OTHERWISE
440 VERT = VERT + 1
450  NORMAL : PRINT
460  VTAB 18
470  PRINT
```

```
480  FOR I = 2 TO 79: PRINT "_";: NEXT I
490  PRINT : PRINT
500  PRINT "Options: ";: INVERSE : PRINT "Q";: NORMAL : PRINT " key to Q
    uit.  "
510  REM PRINT "Press ";
520  REM  INVERSE : PRINT "Open-Apple";: NORMAL : PRINT " and selection
    together, for help."
530  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
540  POKE 36,HRIZ: VTAB VERT: REM  RE-POSITION CURSOR
550 P =  PEEK ( - 16384): REM  GET VALUE FROM KBRD
560  IF P < 127 THEN 540: REM  WAIT UNTIL KEY IS PRESSED
570 P = P - 128:NB$ =  CHR$ (P): PRINT NB$: REM  SUBTRACT 128 FOR ASCII
    VALUE & DISPLAY CHARACTER
580  IF P = 81 OR P = 113 THEN 10000: REM    HOME : PRINT "THAT'S ALL":
    POKE  - 16368,0: END : REM  ESC KEY PRESSED
590  IF  PEEK ( - 16287) > 127 THEN 20000: REM    OPEN-APPLE KEY AND SEL
    ECTION PRESSED
600 NB =  VAL (NB$)
610  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
620  POKE 36,HRIZ + 1: VTAB VERT: REM  RE-POSITION CURSOR
630 P2 =  PEEK ( - 16384):P2 = P2 - 128: REM  GET VALUE FROM KBRD
640  IF P2 = 8 THEN  POKE 36,HRIZ - 1: CALL  - 868: GOTO 530: REM  IF BA
    CKSPACE, CLEAR LINE AND DO AGAIN
650  IF P2 <  > 13 THEN 620: REM  RETURN KEY
660  IF NB < 1 OR NB > 9 THEN  POKE 36,HRIZ - 1: CALL  - 868: VTAB 24:
    INVERSE : HTAB 2:  PRINT "INCORRECT CHOICE! Please Choose Again.
    ";: NORMAL : GOTO      530
670  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
680  IF NB = 1 THEN 1000
690  IF NB = 2 THEN 2000
700  IF NB = 3 THEN 3000
710  IF NB = 4 THEN 4000
720  IF NB = 5 THEN 5000
730  IF NB = 6 THEN 6000
735  IF NB = 7 THEN 7000
740  IF NB = 8 THEN 8000
750  IF NB = 9 THEN 9000
760 :
770 :
780 :
1000  REM  **--OPTION 1--**
1010  HOME
1020  VTAB 5
1025  INVERSE : PRINT "One Moment Please! I'm getting the program.": NORMAL
1030  PRINT D$;"RUN LINE.30.GRAPH"
1980 :
1990 :
2000  REM  **--OPTION 2--**
```

```
2010  HOME
2020  VTAB 5
2030  INVERSE : PRINT "One Moment Please! I'm getting the program."
2040  NORMAL
2050  PRINT D$;"RUN COL.30.GRAPH"
2060  END
2980 :
2990 :
3000  REM  **--OPTION 3--**
3010  HOME
3020  VTAB 5
3025  INVERSE
3030  PRINT "One moment please! I'm getting the program."
3035  NORMAL
3040  PRINT D$;"RUN BAR.30.GRAPH"
3980 :
3990 :
4000  REM  **-OPTION 4--**
4010  HOME
4020  VTAB 5
4025  INVERSE
4030  PRINT "One moment please! I'm getting the program."
4035  NORMAL
4040  PRINT D$;"RUN SCAT.30.GRAPH"
4980 :
4990 :
5000  REM  **--OPTION 5--**
5010  HOME
5020  VTAB 5
5025  INVERSE
5030  PRINT "One moment please! I'm getting the program."
5035  NORMAL
5040  PRINT D$;"RUN DEVIT.15.GRAPH"
5980 :
5990 :
6000  REM   **--OPTION 6--**
6010  HOME
6020  VTAB 5
6025  INVERSE
6030  PRINT "One moment please! I'm getting the program."
6035  NORMAL
6040  PRINT D$;"RUN HI.LW.CLS.GRAPH"
6980 :
6990 :
7000  REM    **--OPTION 7--**
7010  HOME
7020  VTAB 5
7025  INVERSE
```

```
7030  PRINT "One moment please! I'm getting the program."
7035  NORMAL
7040  PRINT D$;"RUN AREA.30.GRAPH"
7980 :
7990 :
8000  REM   **--OPTION 8--**
8010  HOME
8020  VTAB 5
8025  INVERSE
8030  PRINT "One moment please! I'm getting the program."
8035  NORMAL
8040  PRINT D$;"RUN PIE.GRAPH"
8980 :
8990 :
9000  REM   **--OPTION 9--**
9010  HOME
9020  VTAB 5
9025  INVERSE
9030  PRINT "One moment please! I'm getting the program."
9035  NORMAL
9040  PRINT D$;"RUN GRADES.GRAPH"
9980 :
9990 :
10000  REM    **--OPTION 10--**
10010  HOME : POKE  - 16368,0: REM  RESET KEYBOARD
10020  VTAB 5: HTAB 10
10030  INVERSE
10040  PRINT "See you next time."
10050  NORMAL
10060  END
10070 :
10080 :
10090 :
10100 :
30000  REM  **--RETURN SUBROUTINE--**
30010  VTAB 24: PRINT "Press the ";: INVERSE : PRINT "RETURN";: NORMAL :
     PRINT " key to go on to the MENU:";: INPUT " ";NUL$
30020  RETURN
```

Price Graph

```
100  REM   ***--GRAPH.PRICE--***
110 :
120 :
130  REM  **--VARIABLES LIST--**
140  REM  STK$ = STOCK SYMBOL
150  REM  HI$  = CURRENT HI PRICE
```

```
160  REM  LOW$ = CURRENT LOW PRICE
170  REM  VOL  = SALES VOLUME
240 :
250 :
260  REM   **--INITIALIZATION--**
270 D$ =  CHR$ (4): REM  CONTROL D
280  TEXT : PRINT  CHR$ (12): VTAB 10: INVERSE : PRINT "ONE MOMENT PLEAS
    E. I MUST GET SOME INFO.": NORMAL
290  PRINT D$;"OPEN STOCKS,L260"
300  PRINT D$;"READ STOCKS,R0"
310  INPUT PTR
320 REC = PTR
330  PRINT D$;"CLOSE STOCKS"
340  PRINT D$;"BLOAD ASCII.SHP.TABLE": POKE 232,0: POKE 233,96
350 :
360 :
380  REM **--SET UP--**
390  PRINT  CHR$ (12): VTAB 5
400  PRINT D$;"OPEN STOCKS.HI.LOW"
410  PRINT D$;"READ STOCKS.HI.LOW"
420  FOR I = 0 TO 9
430  INPUT STK$(I)
440  INPUT HI$(I)
450  INPUT LOW$(I)
460  PRINT I + 1;".";: HTAB 5: PRINT STK$(I)
470  NEXT I
480 STK$(10) = "STOCK MENU"
490  PRINT "11.";: HTAB 5: PRINT STK$(10)
500  PRINT D$;"CLOSE STOCKS.HI.LOW"
510  PRINT
520  INPUT "WHICH STOCK ";W
530  IF W < 1 OR W > 11 THEN  PRINT "INCORRECT CHOICE": GOTO 520
540  IF W = 11 THEN  PRINT D$;"RUN STK.DISP.MENU"
550 W = W - 1
560 :
570  PRINT : PRINT "BEGIN WITH WHICH RECORD? (1 TO ";REC;") ";: INPUT ""
    ;BR:ER = BR + 49: IF ER > REC THEN ER = REC
580 :
590 :
600 :
610  REM  **--DETERMINE LOWEST PRICE--**
620  IF  VAL (HI$(W)) -  VAL (LOW$(W)) < 210 THEN LOW =  VAL (LOW$(W)): GOTO
    1000
630 LOW =  VAL (HI$(W))
640  PRINT  CHR$ (12): VTAB 10: INVERSE
650  PRINT "PLEASE WAIT. I'M LOOKING FOR THE LOWEST VALUE IN THIS 50 DAY
     PERIOD.": NORMAL
660  PRINT D$;"OPEN STOCKS,L260"
```

```
670 ER = BR + 49
680  IF ER > REC THEN ER = REC
690  FOR K = BR TO ER
700  PRINT D$;"READ STOCKS,R";K;",B";(W * 25) + 10 + 15
710  INPUT DLOW
720  IF DLOW < LOW THEN LOW = DLOW
730  NEXT K
740  GOTO 1000
750 :
760 :
770 :
1000  REM **--DRAW GRAPH--**
1010 G = 1:R = 25
1011  HOME : VTAB 10: PRINT "One moment please. I am drawing the graph."
1012  POKE  - 16297,0: POKE 230,32: CALL  - 3086: REM  DRAW PG.1,CLR SCR
    N
1013  POKE  - 16302,0
1014  GOTO 1018
1015  POKE  - 16297,0
1016  POKE  - 16304,0: POKE  - 16302,0
1017  CALL  - 3086
1018  HCOLOR= 4
1020  HPLOT 0,0: CALL  - 3082
1021  GOTO 1030
1022  FOR SC = 0 TO 191
1024  HPLOT 0,SC TO 279,SC
1026  NEXT SC
1030  HCOLOR= 5
1040  HPLOT R,7 TO R,183
1045  REM  HCOLOR = 7
1050  FOR I = 7 TO 175 STEP 8: HPLOT R,I TO 275,I: NEXT I
1060  REM  HCOLOR = 7
1070  HPLOT R,183 TO 275,183
1072  HCOLOR= 7
1075  FOR I = R + 25 TO 275 STEP 25: HPLOT I,179 TO I,183: NEXT I
1080 :
1090 :
1100 :
1200  REM **--LABEL W/NUMBERS FROM SHAPE TABLE--**
1205  HCOLOR= 7
1210 LOW =  INT (LOW / 10):LOW = LOW * 10
1220  IF LOW < 0 THEN LOW = 0
1230 LOW$ =  STR$ (LOW / 10)
1240 LN =  LEN (LOW$)
1250  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
1260  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( MID$ (LOW$
    ,1,1)):D3 = 32
```

```
1270  IF LN = 3 THEN D1 =  ASC < MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
1280 X = R + 5
1340  SCALE= 1: ROT= 0
1350 Y = 180
1355  IF D1 > 53 THEN 1390
1360  DRAW D1 AT 15,Y
1370  DRAW D2 AT 8,Y
1380  DRAW D3 AT 1,Y
1390 Y = Y - 40
1400  IF Y < 0 THEN 1700
1402  IF D1 = 54 THEN Y = Y + 8
1403  IF D1 = 55 THEN Y = Y + 16
1404  IF D1 = 56 THEN Y = Y + 24
1405  IF D1 = 57 THEN Y = Y + 32
1410 D1 = D1 + 5
1420  IF D1 < 58 THEN 1360
1430 D1 = 48
1440  IF D2 = 32 THEN D2 = 48
1450 D2 = D2 + 1
1460  IF D2 < 58 THEN 1360
1470 D1 = 48:D2 = 48
1480  IF D3 = 32 THEN D3 = 48
1490 D3 = D3 + 1
1500  IF D3 < 58 THEN 1360
1510 :
1520 :
1530 :
1700  REM **--LABEL W/STOCK NAME--**
1710 X = 140:Y = 12
1712  HCOLOR= 4
1715  HPLOT 135,15, TO (135 + ( LEN (STK$(W)) * 10) + 5),15
1717  HCOLOR= 7
1720  FOR T = 1 TO  LEN (STK$(W))
1730 STK =  ASC ( MID$ (STK$(W),T,1))
1760  DRAW STK AT X,Y
1770 X = X + 10
1780  NEXT T
1790 :
1795 :
1800  REM **--LABEL GRAPH--**
1802  DATA   HIGH,53,16,LOW,53,48,CLOSE,53,32,END,0,0
1805  READ LABEL$,X,Y
1806  IF LABEL$ = "END" THEN 1880
1810  HPLOT 46,20 TO 46,50: HPLOT 44,35 TO 48,35
1830  FOR T = 1 TO  LEN (LABEL$)
1840 LB =  ASC ( MID$ (LABEL$,T,1))
1850  DRAW LB AT X,Y
```

```
1860 X = X + 10
1870  NEXT T
1875  GOTO 1800
1880  REM  X=R+3
1890  RESTORE
1900  IF BR < 103 THEN YEAR$ = "1982"
1902  IF BR > 102 THEN YEAR$ = "1983"
1905 X = 140:Y = 168
1910  FOR T = 1 TO  LEN (YEAR$)
1929 YR =  ASC ( MID$ (YEAR$,T,1))
1930  DRAW YR AT X,Y
1940 X = X + 10
1950  NEXT T
1990 X = R + 3
1995  HOME : POKE  - 16304,0
1998 :
1999 :
2000  REM   **--DISK INPUT ROUTINE--**
2005 ALT$ = "BLUE"
2010  PRINT D$;"OPEN STOCKS,L260"
2020  FOR K = BR TO ER
2030  PRINT D$;"READ STOCKS,R";K;",B";0
2040  INPUT DT$
2050  PRINT D$;"READ STOCKS,R";K;",B";(W * 25) + 10
2060  INPUT PE
2070  PRINT D$;"READ STOCKS,R";K;",B";(W * 25) + 10 + 4
2080  INPUT VOL
2090  PRINT D$;"READ STOCKS,R";K;",B";(W * 25) + 10 + 10
2100  INPUT DHIGH
2110  PRINT D$;"READ STOCKS,R";K;",B";(W * 25) + 10 + 15
2120  INPUT DLOW
2130  PRINT D$;"READ STOCKS,R";K;",B";(W * 25) + 10 + 20
2140  INPUT DCLOSE
2150 :
2160 :
2170 :
3000  REM  **--PLOT HI/LOW/CLOSE PRICES--**
3010 DAY = X
3020 Q = DLOW: GOSUB 7000:DLOW = PT
3030  IF DLOW > 191 THEN DLOW = 191
3040  IF DLOW < 0 THEN DLOW = 0
3050  HPLOT DAY,DLOW
3060 Q = DHIGH: GOSUB 7000:DHIGH = PT
3070  IF DHIGH < 0 THEN DHIGH = 0
3080  IF DHIGH > 191 THEN DHIGH = 191
3090  HPLOT DAY,DHIGH
3100 Q = DCLOSE: GOSUB 7000:DCLOSE = PT
3110  IF DCLOSE < 0 THEN DCLOSE = 1
```

```
3120  IF DCLOSE > 191 THEN DCLOSE = 190
3130  HPLOT DAY,DCLOSE
3140  HPLOT DAY,DHIGH TO DAY,DLOW
3145  IF ALT$ = "ORANGE" THEN  HCOLOR= 6:ALT$ = "BLUE": GOTO 3150
3147  IF ALT$ = "BLUE" THEN  HCOLOR= 5:ALT$ = "ORANGE"
3150  HPLOT DAY - 1,DCLOSE TO DAY + 1,DCLOSE
3155  HCOLOR= 7
3170 X = X + 5
3180  IF K = ER THEN  GOSUB 9000
3190  IF G = 5 THEN  GOSUB 8000
3200 G = G + 1
3210  NEXT K
3220  PRINT D$;"CLOSE"
3230 :
3240 :
3250 :
4000  REM **--PAPER PRINT OUT--**
4002  FOR PAUSE = 1 TO 2000: NEXT PAUSE
4003  POKE  - 16303,0: REM  TEXT PAGE
4010  VTAB 10
4030  INPUT "DO YOU WANT A PAPER PRINT OUT ? ";YES$
4040  NORMAL
4050  IF YES$ <  > "Y" THEN 5000
4060  POKE  - 16302,0
4070  POKE 10,76: POKE 11,00: POKE 12,96
4080  PRINT  CHR$ (4);"BLOAD PRINTER.ROUTINE,A$6000"
4090  POKE  - 16297,0: POKE  - 16304,0
4100  PRINT  USR (0101)
4105  PRINT D$;"BLOAD ASCII.SHP.TABLE"
4110  GOTO 5000
4120 :
4130 :
4140 :
5000  REM  **--RETURN TO MENU--**
5005  TEXT
5010  HOME : VTAB 10
5020  INPUT "DO YOU WANT TO SEE THE GRAPH AGAIN? ";YES$
5030  IF YES$ = "Y" THEN 5100
5040  PRINT : PRINT "DO YOU WANT TO SEE ANOTHER GRAPH": PRINT
5050  INPUT "FOR THIS SAME STOCK? ";YES$
5060  IF YES$ = "Y" THEN 5200
5070  GOTO 5510
5100  HOME : VTAB 10
5120  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT " K
    EY WHEN YOU ARE"
5130  PRINT : PRINT "READY TO SEE THE GRAPH."
5140  PRINT : PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY AGAIN WHEN YOU ARE": PRINT : PRINT "FINISHED.";
```

```
5145  INPUT "";L$
5147  HOME
5150  POKE  - 16304,0
5160  INPUT L$
5170  TEXT
5180  PRINT
5190  GOTO 5000
5198 :
5199 :
5200  REM **--DISPLAY MORE--**
5210  HOME : VTAB 10
5220  GOTO 570
5510  PRINT D$;"RUN STK.DISP.MENU"
6996 :
6997 :
6998 :
7000  REM  **--DETERMINE POINTS--**
7010 R =  INT ((Q - LOW) / 10)
7020 R1 = (Q - LOW) - (R * 10)
7030 P1 = (R * 8) + R1
7040 PT = 183 - P1
7050  RETURN
7990 :
7995 :
7998 :
8000  REM  **--DETERMINE DATE--**
8010 I = 1:R = X:X = X - 27
8020  IF  MID$ (DT$,I,1) = "-" OR  MID$ (DT$,I,1) = "/" THEN DASH = I: GOTO
    8050
8030 I = I + 1
8040  GOTO 8020
8050 I = I + 1
8055  IF  MID$ (DT$,I,1) = "-" OR  MID$ (DT$,I,1) = "/" THEN SLASH = I: GOTO
    8080
8060 I = I + 1
8070  GOTO 8055
8080 MNTH$ = ( LEFT$ (DT$,DASH - 1))
8081 :
8082  IF MNTH$ = M2$ THEN X = R:G = 0: RETURN
8085 M2$ = MNTH$
8090 DTE$ = ( MID$ (DT$,(DASH + 1),(SLASH - (DASH + 1))))
8100  FOR T = 1 TO  LEN (MNTH$)
8110 MNTH =  ASC ( MID$ (MNTH$,T,1))
8130  DRAW MNTH AT X,185
8140 X = X + 5
8150  NEXT T
8155  HPLOT X + 1,188: HPLOT X + 2,188
8157 X = X + 4
```

```
8160  FOR T = 1 TO  LEN (DTE$)
8170 DTE =  ASC ( MID$ (DTE$,T,1))
8190  DRAW DTE AT X,185
8200 X = X + 6
8210  NEXT T
8220 X = R
8500 G = 0
8510  RETURN
8520 :
8530 :
8540 :
9000  REM  **--DATE OF LAST RECORD--**
9010  IF G < 5 THEN X = X + 5 * (5 - G)
9020 G = 5
9025 M2$ = ""
9030  RETURN
9998 :
9999 :
10000  REM  STOCK MARKET SYSTEM
11000  REM  COPYRIGHT NOV 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS ROAD
14000  REM  CORVALLIS, OR. 97330
```

Volume Graph

```
100  REM   ***--GRAPH.VOLUME--***
110 :
120 :
130  REM  **--VARIABLES LIST--**
140  REM  STK$ = STOCK SYMBOL
150  REM  HI$  = CURRENT HI PRICE
160  REM  LOW$ = CURRENT LOW PRICE
170  REM  VOL  = SALES VOLUME
230 :
240 :
250 :
260  REM   **--INITIALIZATION--**
270 D$ =  CHR$ (4): REM  CONTROL D
275 LVOL$ = "OFF"
280  TEXT : PRINT  CHR$ (12): VTAB 10: INVERSE : PRINT "ONE MOMENT PLEAS
    E. I MUST GET SOME INFO.": NORMAL
290  PRINT D$;"OPEN STOCKS,L260"
300  PRINT D$;"READ STOCKS,R0"
310  INPUT PTR
320 REC = PTR
330  PRINT D$;"CLOSE STOCKS"
```

```
340  PRINT D$;"BLOAD ASCII.SHP.TABLE": POKE 232,0: POKE 233,96
350 :
360 :
370 :
380  REM **--SET UP--**
390  PRINT  CHR$ (12): VTAB 5
400  PRINT D$;"OPEN STOCKS.HI.LOW"
410  PRINT D$;"READ STOCKS.HI.LOW"
420  FOR I = 0 TO 9
430  INPUT STK$(I)
440  INPUT HI$(I)
450  INPUT LOW$(I)
460  PRINT I + 1;".";: HTAB 5: PRINT STK$(I)
470  NEXT I
480 STK$(10) = "STOCK MENU"
490  PRINT "11.";: HTAB 5: PRINT STK$(10)
500  PRINT D$;"CLOSE STOCKS.HI.LOW"
510  PRINT
520  INPUT "WHICH STOCK ";W
530  IF W < 1 OR W > 11 THEN  PRINT "INCORRECT CHOICE": GOTO 520
540  IF W = 11 THEN  PRINT D$;"RUN STOCK.MENU"
550 W = W - 1
560 :
570  PRINT : PRINT "BEGIN WITH WHICH RECORD? (1 TO ";REC;") ";: INPUT ""
    ;BR:ER = BR + 49: IF ER > REC THEN ER = REC
580 :
590 :
600 :
610  REM  **--DETERMINE LOWEST PRICE--**
640  PRINT  CHR$ (12): VTAB 10: INVERSE
650  PRINT "PLEASE WAIT. I'M LOOKING FOR THE VOL. RANGE IN THIS 50 DAY P
    ERIOD.": NORMAL
660  PRINT D$;"OPEN STOCKS,L260"
670 ER = BR + 49
680  IF ER > REC THEN ER = REC
685 LOW = 99999:HI = 0
690  FOR K = BR TO ER
700  PRINT D$;"READ STOCKS,R";K;",B";(W * 25) + 10 + 4
710  INPUT VOL
720  IF VOL < LOW THEN LOW = VOL
725  IF VOL > HI THEN HI = VOL
730  NEXT K
740  GOTO 1000
750 :
760 :
770 :
1000  REM **--DRAW GRAPH--**
1010 G = 1:R = 21:Y = 177
```

```
1011  HOME : VTAB 10
1012  PRINT "One moment please. I am drawing the graph."
1013  POKE  - 16297,0: REM  INIT HIRES GRAPHICS
1014  POKE  - 16302,0: REM  FULL SCRN
1015  POKE 230,32: REM  DRAW ON PAGE 1
1016  CALL  - 3086: REM  CLEAR SCRN
1017  HCOLOR= 4
1018  HPLOT 0,0
1019  CALL  - 3082: REM  CLEAR TO BLACK2 (4)
1030  HCOLOR= 5
1040  HPLOT R,10 TO R,180
1050  FOR I = 10 TO 180 STEP 10: HPLOT R,I: HPLOT  TO 272,I: NEXT I
1060  HCOLOR= 7
1070  FOR I = 46 TO 272 STEP 25: HPLOT I,178: HPLOT  TO I,182: NEXT I
1071 RANGE = HI - LOW: IF RANGE < 1000 THEN RANGE = RANGE * 10:LVOL$ = "
    ON"
1072 VP = RANGE / 160: IF VP < 10 THEN VP = .05: GOTO 1200
1073 VP =  INT (VP) / 10
1074 VP =  INT (VP + .5)
1075 VP = VP / 10
1080 :
1090 :
1100 :
1200  REM **--LABEL W/NUMBERS FROM SHAPE TABLE--**
1210 LOW =  INT (LOW / 100)
1220  IF LOW < 0 THEN LOW = 0
1225 L1 = LOW
1230 LOW$ =  STR$ (LOW)
1240 LN =  LEN (LOW$)
1250  IF LN = 1 THEN D1 =  ASC (LOW$):D2 = 32:D3 = 32
1260  IF LN = 2 THEN D1 =  ASC ( MID$ (LOW$,2,1)):D2 =  ASC ( LEFT$ (LOW
    $,1)):D3 = 32
1270  IF LN = 3 THEN D1 =  ASC ( MID$ (LOW$,3,1)):D2 =  ASC ( MID$ (LOW$
    ,2,1)):D3 =  ASC ( LEFT$ (LOW$,1))
1280 X = R + 5
1340  SCALE= 1: ROT= 0
1350  IF D1 > 53 THEN LOW = LOW + (VP * 10):Y = Y - 10: GOTO 1230
1360  DRAW D1 AT 14,Y
1370  DRAW D2 AT 7,Y
1380  DRAW D3 AT 0,Y
1390 Y = Y - 50
1400  IF Y < 0 THEN 1700
1402  IF D1 = 54 THEN Y = Y + 10
1404  IF D1 = 55 THEN Y = Y + 20
1406  IF D1 = 56 THEN Y = Y + 30
1408  IF D1 = 57 THEN Y = Y + 40
1410 LOW = LOW + ((VP * 5) * 10)
1420  GOTO 1230
```

```
1510 :
1520 :
1530 :
1700  REM **--LABEL W/STOCK NAME--**
1705  IF BR < 103 THEN YEAR$ = "1982"
1706  IF BR > 102 THEN YEAR$ = "1983"
1710 X = 125:Y = 0
1720  FOR T = 1 TO  LEN (STK$(W))
1730 STK =  ASC ( MID$ (STK$(W),T,1))
1760  DRAW STK AT X,Y
1770 X= X + 10
1780  NEXT T
1795  HOME : POKE  - 16304,0: REM  SWITCH TO GRAPHIC DISPLAY
1800 X = 0:Y = 0
1810 LABEL$ = "VOL."
1820  FOR T = 1 TO  LEN (LABEL$)
1830 L9 =  ASC ( MID$ (LABEL$,T,1))
1840  DRAW L9 AT X,Y
1850 X = X + 10
1860  NEXT T
1865  IF LABEL$ = "VOL." THEN LABEL$ = YEAR$:X = 239:L9 = 0: GOTO 1820
1870 X = R + 3
1895  HOME : POKE  - 16304,0: REM   SWITCH TO GRAPHC DISPLAY
1990 :
1995 :
2000  REM  **--DISK INPUT ROUTINE--**
2010  PRINT D$;"OPEN STOCKS,L260"
2020  FOR K = BR TO ER
2030  PRINT D$;"READ STOCKS,R";K;",B";0
2040  INPUT DT$
2070  PRINT D$;"READ STOCKS,R";K;",B";(W * 25) + 10 + 4
2080  INPUT VOL
2090  IF LVOL$ = "ON" THEN VOL = VOL * 10
2150 :
2160 :
2170 :
3000  REM   **--PLOT VOLUME--**
3010 DAY = X
3020  GOSUB 7000
3030  HCOLOR= 6
3130  HPLOT DAY,PT
3140  HPLOT DAY,PT TO DAY,180
3150  HPLOT DAY - 1,PT TO DAY - 1,180
3170 X = X + 5
3175  HCOLOR= 7
3180  IF K = ER THEN  GOSUB 9000
3190  IF G = 5 THEN  GOSUB 8000
3200 G = G + 1
```

```
3210  NEXT K
3220  PRINT D$;"CLOSE"
3230 :
3240 :
3250 :
4000  REM **--PAPER PRINT OUT--**
4002  FOR PAUSE = 1 TO 2000: NEXT PAUSE
4003  POKE  - 16303,0: REM  TEXT PAGE
4010  VTAB 10
4030  INPUT "DO YOU WANT A PAPER PRINT OUT ? ";YES$
4040  NORMAL
4050  IF YES$ <  > "Y" THEN 5000
4060  POKE  - 16302,0
4070  POKE 10,76: POKE 11,00: POKE 12,96
4080  PRINT  CHR$ (4);"BLOAD PRINTER.ROUTINE,A$6000"
4090  POKE  - 16297,0: POKE  - 16304,0
4100  PRINT  USR (0101)
4105  PRINT D$;"BLOAD ASCII.SHP.TABLE"
4110  GOTO 5000
4120 :
4130 :
4140 :
5000  REM  **--RETURN TO MENU--**
5002  HOME
5005  TEXT
5010  HOME : VTAB 10
5020  INPUT "DO YOU WANT TO SEE THE GRAPH AGAIN? ";YES$
5030  IF YES$ = "Y" THEN 5100
5040  PRINT : PRINT "DO YOU WANT TO SEE ANOTHER GRAPH": PRINT
5050  INPUT "FOR THIS SAME SUBJECT? ";YES$
5060  IF YES$ = "Y" THEN 5200
5070  GOTO 5510
5100  HOME : VTAB 10
5120  PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT " K
    EY WHEN YOU ARE"
5130  PRINT : PRINT "READY TO SEE THE GRAPH."
5140  PRINT : PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT
    " KEY AGAIN WHEN YOU ARE": PRINT : PRINT "FINISHED.";
5145  INPUT "";L$
5147  HOME
5150  POKE  - 16304,0
5160  INPUT L$
5170  TEXT
5180  PRINT
5190  GOTO 5000
5198 :
5199 :
5200  REM **--DISPLAY MORE--**
```

```
5210  HOME : VTAB 10
5220  GOTO 570
5510  PRINT D$:"RUN STK.DISP.MENU"
6996 :
6997 :
6998 :
7000  REM  **--DETERMINE POINTS--**
7010 P1 =  INT (VOL / 100 + .5)
7012 P2 = P1 - L1
7014 P3 =  INT (P2 / VP)
7020 PT = 180 - P3
7050  RETURN
7990 :
7995 :
7998 :
8000  REM  **--DETERMINE DATE--**
8010 I = 1:R = X:X = X - 27
8020  IF  MID$ (DT$,I,1) = "-" OR  MID$ (DT$,I,1) = "/" THEN DASH = I: GOTO
    8050
8030 I = I + 1
8040  GOTO 8020
8050 I = I + 1
8055  IF  MID$ (DT$,I,1) = "-" OR  MID$ (DT$,I,1) = "/" THEN SLASH = I: GOTO
    8080
8060 I = I + 1
8070  GOTO 8055
8080 MNTH$ = ( LEFT$ (DT$,DASH - 1))
8082  IF MNTH$ = M2$ THEN X = R:G = 0: RETURN : REM  REMOVE TO DISPLAY E
    VERY WEEK
8085 M2$ = MNTH$
8090 DTE$ = ( MID$ (DT$,(DASH + 1),(SLASH - (DASH + 1))))
8100  FOR T = 1 TO  LEN (MNTH$)
8110 MNTH =  ASC ( MID$ (MNTH$,T,1))
8130  DRAW MNTH AT X,184
8140 X = X + 5
8150  NEXT T
8155  HPLOT X + 1,187: HPLOT X + 2,187
8157 X = X + 4
8160  FOR T = 1 TO  LEN (DTE$)
8170 DTE =  ASC ( MID$ (DTE$,T,1))
8190  DRAW DTE AT X,184
8200 X = X + 6
8210  NEXT T
8220 X = R
8500 G = 0
8510  RETURN
8520 :
8530 :
```

```
8540 :
9000  REM  **--DATE OF LAST RECORD--**
9010  IF G < 5 THEN X = X + 5 * (5 - G)
9020 G = 5
9025 M2$ = ""
9030  RETURN
9998 :
9999 :
10000  REM  STOCK MARKET SYSTEM
11000  REM  COPYRIGHT NOV 1987
12000  REM  DAVID H. MILLER
13000  REM  1750 SULPHUR SPRINGS ROAD
14000  REM  CORVALLIS, OR. 97330
```

Stock System Menu

```
100  REM   ***--STK.DISP.MENU--***
110 :
120 :
130 TB = 2
140 D$ =  CHR$ (4): REM  CONTROL D
150  DIM A(20)
160 :
170 :
180  REM  **--DISPLAY STOCK INFO. MENU--**
190  HOME : REM  PRINT D$;"PR#3"
200  FOR I = 2 TO 79: PRINT "_";: NEXT I
210  PRINT
220  VTAB 3
230  HTAB TB
240  PRINT "STOCK MARKET";
250 TITLE$ = "DISPLAY STOCK INFO. MENU"
260 LN =  LEN (TITLE$)
270 CENTER = (80 - LN) / 2: REM  CENTER TITLE
280  HTAB CENTER
290  INVERSE
300  PRINT TITLE$;
310  NORMAL
320  PRINT  SPC( 10): REM   10 SPACES
330  PRINT "Select Action"
340  PRINT
345  PRINT : HTAB TB: PRINT "1.  Numeric Information (Text)
         6.  List of Diskette Files"
350  PRINT : HTAB TB: PRINT "2.  Graphic Display of Price
         7.  Return to the System Menu"
360  PRINT : HTAB TB: PRINT "3.  Graphic Display of Volume"
```

```
370  PRINT : HTAB TB: PRINT "4.  Hi/Low Information"
380  PRINT : HTAB TB: PRINT "5.  Not Yet Used"
390  PRINT : PRINT
410  VTAB 18: HTAB TB: PRINT "Type your Selection ";: INVERSE : PRINT "(
    1-7)";: NORMAL : PRINT " and press ";: INVERSE : PRINT "RETURN";: NORMAL
    : PRINT ";";
420  NORMAL : PRINT " ---> ";
430 VERT = PEEK (37):HRIZ =  PEEK (1403); REM  $37 CONTAINS VERTICAL CU
    RSOR POSITION; $1403 CONTAINS HORIZONTAL CURSOR POSITION FOR 80 COL.
    //e--$36 OTHERWISE
440 VERT = VERT + 1
450  NORMAL : PRINT
460  VTAB 18
470  PRINT
480  FOR I = 2 TO 79: PRINT "_";: NEXT I
490  PRINT : PRINT
500  PRINT "Options: ";: INVERSE : PRINT "Q";: NORMAL : PRINT "key to l
    eave.  "
510  REM  PRINT "Press ";
520  REM  INVERSE : PRINT "Open-Apple";: NORMAL : PRINT " and selection
    together, for help."
530  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
540  POKE 36,HRIZ: VTAB VERT: REM  RE-POSITION CURSOR
550 P =  PEEK ( - 16384): REM  GET VALUE FROM KBRD
560  IF P < 127 THEN 540: REM  WAIT UNTIL KEY IS PRESSED
570 P = P - 128:NB$ =  CHR$ (P): PRINT NB$: REM  SUBTRACT 128 FOR ASCII
    VALUE & DISPLAY CHARACTER
580  IF P = 81 OR P = 113 THEN  HOME : PRINT "THAT'S ALL": POKE  - 16368
    ,0: END : REM     Q KEY PRESSED
590  IF  PEEK ( - 16287 ) > 127 THEN 20000: REM   OPEN-APPLE KEY AND SELE
    CTION PRESSED
600 NB =  VAL (NB$)
610  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
620  POKE 36,HRIZ + 1: VTAB VERT: REM  RE-POSITION CURSOR
630 P2 =  PEEK ( - 16384):P2 = P2 - 128: REM  GET VALUE FROM KBRD
640  IF P2 = 8 THEN  POKE 36,HRIZ - 1: CALL  - 868: GOTO 530: REM  IF BA
    CKSPACE, CLEAR LINE AND DO AGAIN
650  IF P2 <  > 13 THEN 620: REM  RETURN KEY
660  IF NB < 1 OR NB > 7 THEN  POKE 36,HRIZ - 1: CALL  - 868: VTAB 24: INVERSE
    : HTAB 2: PRINT "INCORRECT CHOICE! Please Choose Again. ";: NORMAL : GOTO
    530
670  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
680  IF NB = 1 THEN 1000
690  IF NB = 2 THEN 2000
700  IF NB = 3 THEN 3000
710  IF NB = 4 THEN 4000
720  IF NB = 5 THEN 5000
730  IF NB = 6 THEN 6000
```

```
735  IF NB = 7 THEN 7000
740 :
750 :
1000  REM  **--OPTION 1--**
1010  HOME
1020  VTAB 5
1025  INVERSE : PRINT "One Moment Please! I'm getting the program.": NORMAL

1030  PRINT D$;"RUN DISPLAY.STOCKS"
1040  END
1980 :
1990 :
2000  REM  **--OPTION 2--**
2010  HOME
2020  VTAB 5
2030  INVERSE : PRINT "One Moment Please! I'm getting the program."
2040  NORMAL
2050  PRINT D$;"RUN GRAPH.PRICE"
2060  END
2980 :
2990 :
3000  REM  **--OPTION 3--**
3010  HOME
3020  VTAB 5
3025  INVERSE
3030  PRINT "One moment please! I'm getting the program."
3035  NORMAL
3040  PRINT D$;"RUN GRAPH.VOLUME"
3980 :
3990 :
4000  REM **--OPTION 4--**
4010  HOME
4020  VTAB 5
4025  INVERSE
4030  PRINT "One moment please! I'm getting the Hi/Low Display Program."

4035  NORMAL
4040  PRINT D$;"RUN DISPLAY.HI.LOW"
4050  GOTO 180: REM  MENU
4980 :
4990 :
5000  REM   **--OPTION 5--**
5010  HOME
5020  VTAB 5
5025  INVERSE
5030  PRINT "NOT USED YET!"
5035  NORMAL
5040  GOSUB 30000: REM  RETURN SUBROUTINE
```

```
5050  GOTO 180: REM  MENU
5980 :
5990 :
6000  REM  **--OPTION 6--**
6010  HOME
6020  PRINT
6030  PRINT D$;"CATALOG"
6035  NORMAL
6040  PRINT
6050  GOSUB 30000: REM  RETURN SUBROUTINE
6060  GOTO 180: REM  MENU
6980 :
6990 :
7000  REM  **--OPTION 7--**
7010  HOME
7020  VTAB 5
7030  INVERSE
7040  PRINT "One moment please! I'm getting the program."
7050  NORMAL
7060  PRINT D$;"RUN STOCK.MENU"
7980 :
7990 :
9980 :
9990 :
30000  REM  **--RETURN SUBROUTINE--**
30010  VTAB 24: PRINT "Press the ";: INVERSE : PRINT "RETURN";: NORMAL :
     PRINT " key to go on to the MENU:";: INPUT " ";NUL$
30020  RETURN
```

CHAPTER 10

IIGS Graphics

Using Applesoft BASIC to create graphic displays on the Apple IIGS super high-resolution screens is one of the more interesting tasks I have encountered since becoming involved with computers. Most programming is a matter of learning the language and logic of a specific system and applying that knowledge to get a desired result. The choices that must be made are often dictated by the hardware, the operating system or the programming language. The Apple IIGS seems to present a unique and sometimes schizophrenic situation in which the hardware, operating system, and language are designed to aid the programmer, yet in the ways they work together (or do not work together) they often create frustration and make it more difficult to accomplish certain tasks.

A brief example may help explain the situation. As I explained in Chapter 4, Apple did not update Applesoft BASIC to allow BASIC programmers direct access to the super high-resolution screen or to the many ROM routines stored inside the IIGS computer. Applesoft itself, therefore, would seem to prohibit BASIC programmers from ever using that language to create super high-resolution graphic displays. However, Apple did include a mechanism called "shadowing" that allows information stored in one section of memory to be duplicated in another section of memory. Since BASIC has access to that first section of memory, it *can* be used to produce graphic displays (of a somewhat limited nature) on the IIGS super high-resolution screens in either 320 or 640 mode. On the one hand, Apple appears to be preventing BASIC programmers from using the super high-resolution screen while on the other hand they have provided a way for them to do just that.

IIGS MEMORY

To use the IIGS super high-resolution graphic screen, you must have an understanding of at least some of the memory arrangement on the IIGS. At this point, a caution is in order. Some of the following material goes beyond Applesoft BASIC programming and involves

assembly language and memory management. I have tried to explain everything in a way that is understandable, but I acknowledge that some of the information is rather complex and not necessarily material for the beginner. You may wish to skip the explanation and simply study and/or use the program listings included at the end of this chapter.

As you may recall from Chapter 4, the IIGS is a 16-bit machine, capable of processing 16 pieces of information at one time. All other Apple *II* computers are 8-bit machines, capable of handling only eight pieces of information at a time. This difference imposes a limitation on the amount of memory that can be accessed directly. The Apple IIGS has a working limitation of approximately eight million characters of memory that can be directly accessed (and a much higher theoretical limitation). Earlier Apple *II*'s have an actual limitation of approximately 65 thousand characters of memory (64K) that can be directly accessed (Apple and third-party suppliers have devised ways of using 8-bit computers with more than 64K by "bank selecting" or "bank switching" additional 64K sections of memory, but the maximum that can be directly accessed is still 64K).

These differences in memory limitations are readily demonstrated in the manner by which memory can be examined. All Apple *II* computers except the IIGS require a maximum of four hexadecimal digits to designate a memory location. IIGS computers require up to six hexadecimal numbers to designate memory locations. (Once a bank is designated, though, only four numbers are necessary until access to memory locations outside the bank is desired.) In other words, it requires more numbers to pinpoint a specific memory location on the Apple IIGS.

Even though the IIGS has a much larger directly accessible memory than other versions of the Apple *II* computer, the IIGS memory is still arranged in 64K banks (parts or sections) of memory. Bank 00 coincides with the memory in all other Apple *II* computers. Bank 01 coincides with the additional memory (called Auxiliary memory) available in enhanced Apple *II*e and *II*c computers. In addition, all Apple IIGS's have two other banks available with the possibility of adding up to a total of eight million characters of memory. The third part or memory bank is E0 and the fourth is E1. With 64K available in each section, the minimum total memory available on IIGS computers at this time is 256K.

Bank	*Amount of Memory*	*Running Total*
00	64K	64K
01	64K	128K
E0	64K	192K
E1	64K	256K

The numbers in each section of memory go from $0000 to $FFFF or, in decimal values, from 0 to 65535. Therefore, the following format is required to address any single memory location:

Bank #/Address within that bank

For example,

E1/2000

identifies the bank location as the fourth section of memory and the address within that bank as $2000.

The problem for BASIC programmers is that Applesoft only works in the first bank or section of memory (00), while super high-resolution memory is located in bank E1. It appears then that Applesoft BASIC programmers cannot access super high-resolution graphics. But, in fact, Apple provided a standard way for programmers to access the Auxiliary (bank 01) memory on enhanced Apple *II*e and Apple *II*c computers. Then, in the IIGS, they allowed memory banks 00 and 01 to be "shadowed" or copied into banks E0 and E1 through the use of a POKE to a location in the first bank of memory. In addition, as you saw in Chapter 4, another location in bank 00 allows the BASIC programmer to switch the screen display to super high-resolution. Therefore, those using BASIC do have a path (although at this point it is a very narrow one) through which they can make use of some of the high-resolution graphic capability of the IIGS.

A POKE to location 49205 with the value 0 initiates shadowing of banks 00 to E0 and 01 to E1. You saw in Chapter 4 that using a POKE to location 49193 with the decimal value 163 allows you to switch the screen display to super high-resolution.

```
POKE 49205,0 {RETURN}
POKE 49193,163 {RETURN}
```

To make use of these instructions, you must be able to transfer information from the first 64K section of memory (bank 00) to the second section of memory (bank 01). When the necessary information has been transferred to bank 01 and shadowing is turned on, the information is also copied (automatically shadowed) into bank E1, the bank containing the super high-resolution screen memory. The hard part is getting the information from bank 00 to bank 01.

MEMORY MOVER: AN ASSEMBLY-LANGUAGE ROUTINE

The reference manuals for the Apple *II*e explain how memory in bank 00 can be transferred to bank 01, while at the same time warning BASIC programmers not to use the method: "Do not attempt to use the auxiliary memory from a BASIC program." (Apple *II*e Reference Manual, p. 71, Apple Computer Inc., 1982, and Apple *II*e Technical Reference Manual, p. 86, Addison-Wesley Publishing Company, Inc., 1986.)

Despite this warning, information can be transferred from bank 00 to bank 01 (auxiliary memory) using a BASIC program that supplies an assembly-language routine with certain necessary addresses and then instructs that assembly-language routine to transfer the information. A routine in ROM (called AUXMOVE) is designed to move information between main memory (bank 00) and auxiliary memory (bank 01). The

programmer must supply this routine with the main memory location of the information to be transferred and the auxiliary memory location to transfer it to. Therefore, the structure of the assembly-language routine that moves memory from 00 to 01 is

1. Provide the starting address of the information to be moved
2. Provide the ending address of the information to be moved
3. Provide the starting address for the destination of the information
4. Tell the AUXMOVE routine to actually make the transfer

Listed below is a short assembly language routine that accomplishes these four tasks:

```
300:A900       LDA #00
302:853C       STA 3C
304:A920       LDA #20
306:853D       STA 3D
308:A901       LDA #01
30A:853E       STA 3E
30C:A920       LDA #20
30E:853F       STA 3F
310:A9E0       LDA #E0
312:8542       STA 42
314:A93A       LDA #3A
316:8543       STA 43
318:38         SEC
319:20 11 C3   JSR C311
31C:60         RTS
```

There are three ways you can enter this routine.

1. You can leave BASIC with the command CALL-151 and then enter the addresses and the hexadecimal values (probably the easiest way for such a short routine)
2. you can use the built-in mini-assembler to enter the assembly-language instructions (the LDA, STA, etc. commands)
3. you can type in the following BASIC program that uses the commands DATA, READ, and POKE to enter the routine.

```
100 REM ***--MOVE.MEMORY--***
110:
120:
130 REM **--POKE ROUTINE--**
140 FOR L = 768 TO 796
150 READ V
160 POKE L,V
170 NEXT L
180:
```

```
190:
200 REM **--END--**
210 END
220:
230:
500  REM **--ROUTINE DATA--**
510:
520 REM *--START ADDRESS--*
530 DATA 169,0
540 DATA 133,60
550 DATA 169,32
560 DATA 133,61
570:
580 REM *--ENDING ADDRESS--*
590 DATA 169,1
600 DATA 133,62
610 DATA 169,32
620 DATA 133,63
630:
640 REM *--DESTINATION ADDRESS--*
650 DATA 169,224
660 DATA 133,66
670 DATA 169,58
680 DATA 133,67
690:
700 DATA 56:REM SET CARRY BIT
710:
720 DATA 32,17,195:REM JUMP TO AUXMOVE AT C311
730:
740: DATA 96:REM RTS
```

Starting Address

I presented the routine in assembly language first, because I want to explain it using hexadecimal addresses. Remember that the first thing is to provide the starting address of the information to be moved. The address of the start of the first page of high-resolution memory is $2000. For reasons that will be explained later, the high-resolution area of memory is the area that will be used as the location for the information that will be moved.

The address for the start of High-Resolution Page 1, $2000, is given to the computer with the last two digits first (00—called the low-order byte) and the first two digits last (20—called the high-order byte). This starting address must be given to the computer in specific locations of the computer's memory—locations 3C and 3D. Location 3C gets the low-order byte of the address (00), while location 3D gets the high-order byte of the address (20). So the important numbers to remember are in the second column of the assembly-language routine:

00
3C
20
3D

Memory location 3C is to receive the low-order byte (00) of the starting address of the information to be transferred while memory location 3D is to receive the high-order byte (20) of the starting address of the information to be transferred. The first column contains the hexadecimal codes for the assembly-language instructions telling the computer to place the indicated values in the indicated memory locations (A9 is the hex code for the assembly instruction LDA; 85 is the hex code for the assembly instruction STA). Therefore, the first eight hexadecimal numbers or bytes accomplish our first task of providing the starting address of the information that is to be transferred from bank 00 to bank 01.

Ending Address

The second set of eight hexadecimal numbers in the second column accomplish the second task of providing the ending address of the information that is to be transferred:

01
3E
20
3F

Memory location 3E is to receive the low-order byte (01) of the ending address of the information to be transferred while memory location 3F is to receive the high-order byte (20) of the ending address of the information to be transferred. The first column, A9 85 A9 85, again simply provides the hexadecimal code for the instructions telling the computer to store the indicated values in the indicated addresses.

So far we have told the computer that we want to move two pieces of information, i.e., the first two bytes of the information stored in High-Resolution Page 1, memory locations \$2000 and \$2001. If we were going to move nine pieces of information, then the hexadecimal value 01 would have to be changed to a hexadecimal value of 09. If we wanted to move 15 pieces of information then the 01 would become 0F, etc.

Destination Address

The third group of eight hexadecimal numbers accomplishes the third task: provide the starting address for the destination of the information to be transferred. Once again, the second column contains the important figures: E0 42 3A 43. The destination address is

$3AE0. Memory location 42 gets the low-order byte (E0) while memory location 43 gets the high-order byte (3A) of this example destination address.

We now have provided the computer with the beginning and ending addresses of the information we want to transfer (we have defined the area of memory to be moved) and informed the computer where the information is to be placed in bank 01. One last step remains.

AUXMOVE

The computer must now be told to go ahead and move the information. Apple added flexibility by allowing memory to be moved in either direction, so before it can move the information, the computer must be told in which direction to make the move. Hexadecimal code 38 (assembly instruction SEC) indicates to the computer that the move is to be made from main memory (00) to auxiliary memory (01) and not the other way around. Once the direction is established, we can tell the computer to go to the AUXMOVE routine (located at memory location $C311) and actually make the transfer. The assembly-language routine concludes just like a GOSUB routine in BASIC, with a RETURN (RTS—return from subroutine) hexadecimal code 60.

I have spent a good deal of time explaining this assembly-language routine because it is critical to using BASIC to create graphics on the super high-resolution screen. You should review this explanation until you clearly understand how it works and how information is transferred from bank 00 to bank 01.

I positioned this assembly-language routine at location $300, because it is short enough to fit in the area left free by Apple for just such routines. The routine should be saved on diskette with the following command:

```
BSAVE MEMORY MOVER,A$300,L$1D {RETURN}
```

Within a BASIC program, the routine can be called upon with the command:

```
CALL 768
```

If a numeric variable has been set to equal the value of 768 (for example, MM = 768) then the command can take the form:

```
CALL MM
```

All these commands are demonstrated in the programs listed at the end of the chapter.

SUPER HIGH-RESOLUTION GRAPHICS REQUIREMENTS

Now that you know how to move information from bank 00 to bank 01 (CALL 768), shadow bank 01 to bank E1 (POKE 49205,0), and display the super high-resolution screen located in bank E1 (POKE 49193,163), it would seem possible to immediately display graphics on the super high-resolution screen by using the BASIC high-resolution commands. Theoretically, you should be able to

1. Draw a picture on the high-resolution screen (bank 00)
2. Call the memory-mover routine and move the picture to bank 01
3. Turn on shadowing so that the picture is also copied into bank E1
4. Switch the screen display to super high-resolution
5. Display a super high-resolution graphic image

Although a variation of this sequence eventually will produce super high-resolution graphics, the process is not that simple.

Apple significantly altered the way graphics are created when it introduced super high-resolution. In both low-resolution and high-resolution graphics, the programmer makes choices from the available options: Low-resolution or high-resolution? Which of the 16 low-resolution colors should be used?, etc. The choices in each mode are made within specifically defined limitations: 16 colors or 8 colors, 40 columns or 280 columns, etc. Super high-resolution graphics does not have the same limitations. In fact the programmer is required to define the limitations before using super high-resolution.

Increased flexibility and power require increased knowledge and responsibility on the programmer's part. Super high-resolution is very flexible and very powerful and, therefore, requires a good deal of knowledge and careful programming. Essentially, the programmer is responsible for defining the colors that can be used on the super high-resolution screen and defining the resolution of *each line* on that screen. These two tasks have been identified by Apple as (1) creating color tables, and (2) creating scanline control bytes (SCB's). We will begin by looking at the second task.

Scanline Control Bytes

The SCB's are the definitions for the resolution of each horizontal line of the super high-resolution screen. Each of the 200 lines on the screen can be in either the 320-resolution mode or the 640-resolution mode and both modes can be used on the super high-resolution screen at the same time. Half the screen can be in the 320 mode and the other half in the 640 mode. Any combination the programmer wants is possible, but the definitions for the 200 lines must be established before anything can go on the screen. Once defined, the definitions can be changed within a program as many times as the programmer

desires. The main requirement is that some type of definition for each line must be made prior to the display of super high-resolution graphics.

SCB Definitions

In many ways, creating SCB's is similar to issuing a GR or HGR command to inform the computer to go into low-resolution or high-resolution mode. The difference is that instead of two separate areas of memory, each dedicated to a specific type of resolution, super high-resolution is just one large area of memory, and the programmer decides what the resolution of each line should be.

The 320 mode requires a definition of 0, and 640 mode requires a decimal value definition of 128. (Other values are possible but for now this definition is sufficient.) Chapter 11 presents a more complete discussion of the different possible definitions. The definitions for each line (the scanline control bytes) are stored in memory bank E1 beginning at location $9D00. Each line has its own definition location, so there are 200 bytes used for line definitions. All 200 lines can be defined with the same resolution. By using a BASIC loop, it is not hard to define or set the SCB's for super high-resolution. We begin the super high-resolution demo program with a subroutine that enters a single value for all 200 lines. Type in the following:

```
7000 REM **--SCANLINE CONTROL BYTES--**
7010 FOR I = 0 TO 199
7020 POKE 8192 + I,M
7030 NEXT I
7040 RETURN
7050:
7060:
```

If M is set to equal 0, each super high-resolution line will be defined for 320-resolution mode. If M equals 128 then each line will be defined for 640 mode.

Notice though, that with the loop, you are placing the value of either 0 or 128 in the first 200 bytes of High-Resolution Page 1 (8192 + I, where the value of I goes from 0 to 199, and 8192 is the decimal value for the start of High-Resolution Page 1). Those 200 pieces of information must be transferred to bank E1 using the memory-mover routine.

Transferral of SCB Definitions to Super High-Resolution Memory

If you provide the memory-mover routine with the starting address of the information, the ending address of the information, and the destination for the information, you can transfer the information from bank 00 to bank 01 and shadow it into bank E1 where it really belongs. In order to define and then move the 200 SCB's the following code should be used:

```
700 **--SUPER RES INITIALIZATION--**
710 POKE 49205,0:REM INITIALIZE SHADOWING
```

```
720 MM = 768:REM LOCATION OF ASSEMBLY ROUTINE
730:
740:
800 Rem **--320 MODE--**
805 M = 0: REM 320 MODE
810 GOSUB 7000:REM POKE SCB'S
815 POKE 49193,163:REM SUPER RES
820 EH = 32:EL = 199:DH = 157:DL = 0:GOSUB 17000:REM MOVE SCB'S

17000 REM **--MEMORY MOVER ADDRESSES--**
17010 POKE 777,EL:REM ENDING LOW BYTE
17020 POKE 781,EH:REM ENDING HIGH BYTE
17030 POKE 785,DL:REM DESTINATION LOW BYTE
17040 POKE 789,DH:REM DESTINATION HIGH BYTE
17050 CALL MM
17060 RETURN
17070:
17080:
```

It is important to turn on the super high-resolution mode (815) *before* moving the SCB's or the color-table information.

Please notice that line 820 and the routine at line 17000 do not contain values for the starting address. The starting address remains the same as before, $2000 and therefore does not need to be changed. But the values in 820 suggest an important point to remember.

The addresses for the ending address and the destination address (especially the destination address) can be changed by POKEing new values into the memory-mover routine and then calling that routine again. Because the IIGS can operate so fast, POKEing new addresses for the memory-mover routine and then calling that routine is a practical solution to some difficult problems associated with placing graphics on the super high-resolution screen. At this point, the important thing to remember is that the addresses in MEMORY MOVER can be changed whenever you need to make a change.

Also note that, now that you are working with BASIC, you must use decimal numbers not hexadecimal numbers. I could have included a hexadecimal to decimal conversion routine, but the more calculations the computer must perform, the slower it will go. And under these conditions, speed is a factor. Therefore, instead of using hexadecimals and converting them, I have used their decimal equivalents:

Hex	*Decimal*	*Address*
300	768	
301	769	Starting, low-order byte
302	770	
303	771	
304	772	

305	773	Starting, high-order byte
306	774	
307	775	
308	776	
309	777	Ending, low-order byte
30A	778	
30B	779	
30C	780	
30D	781	Ending, high-order byte
30E	782	
30F	783	
310	784	
311	785	Destination start, low-order byte
312	786	
313	787	
314	788	
315	789	Destination start, high-order byte
316	790	

The designated addresses are the locations that are important and that can be altered.

In the subroutine at line 17000, you change the original assembly-language values by POKEing new ending and destination addresses into the correct memory locations and then calling upon the routine to transfer the 200 bytes stored between $2000 (8192 decimal) and $20C7 (8391 decimal) to bank 01 memory location 9D00 (40192 decimal). Eventually you will use a subroutine that calculates the low-order and high-order byte values from a decimal address such as 40192, but for now all you need to know is that $9D equals 157 decimal, $C7 equals 199 decimal, and $20 equals 32 decimal. Thus the values assigned in line 820:

```
EH=32:EL = 199:DH = 157:DL = 0:GOSUB 17000:REM MOVE SCB'S
```

By using BASIC instructions and a short assembly-language routine, you have defined (set the characteristics of) each line on the super high-resolution screen. The first major task in preparing to use super high-resolution is now finished. The SCB's have been set.

Color Tables

In low- and high-resolution graphics, the number of colors was preset and defined by Apple. With super high-resolution, the number of colors, as well as the characteristics of the colors, can all be established by the programmer. This gives you a lot of flexibility, but also requires you to know what you are doing to take advantage of that flexibility.

Color-Table Definitions

Super high-resolution allows up to 16 color tables in memory at one time, with each color table composed of 16 colors defined by 32 bytes of information. A color is defined by turning 16-bit patterns on or off and storing the resulting two-byte values in the color table. The first four bits (0–3) determine the intensity of blue in the color, the second four bits (4–7) determine the intensity of green in the color, and the third four bits (8–11) determine the intensity of red in the color. The last four bits are reserved and must be set to off or zero. Therefore a color can be defined as follows:

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value	0	0	0	1	1	1	0	1	1	1	1	1	0	0	0	0
Hex	1				D				F				0			

Arranged properly, the definition reads

OF   D1

Translated to decimal values, the definition becomes

15   209

As usual, the low-order byte must be entered first in the color table. The color-table entry for this example color would read

209,15

(Notice, we have just performed a double reverse – confusing, isn't it?) Sixteen colors can be defined in each of the 16 possible color tables. I include only two color tables as examples in this chapter's programs. Experimenting with color definition can produce some surprising results.

To enter the color-table information, the following BASIC code should be added to the example program:

```
825 GOSUB 7500:REM POKE COLOR TABLE
830 EH = 32:EL = 31:DH = 158:DL = 0:GOSUB 17000:REM MOVE 320 COLOR
    TABLE
7500 REM **--COLOR TABLE--**
7510 FOR I = 0 TO 31
7520 READ CT
7530 POKE 8192 + I,CT
7540 NEXT I
7550 RETURN
7560:
7570:
```

```
9000 REM **--320 MODE STANDARD COLOR TABLE--**
9010 DATA 0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020 DATA 169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9030:
9040:
```

The 16 color definitions (each requiring two bytes) are read into the numeric variable CT and are then POKEd into the high-resolution screen memory (8192 + I).

Transferral of Color-Table Definitions to Super High-Resolution Memory

After all 32 bytes are entered, the computer returns to the line following the GOSUB instruction, line 830, where the information that is to be transferred is specified. The ending address of the color table is $201F (or 32 decimal for the ending high-order byte and 31 for the ending low-order byte).

There can be 16 color tables in memory at one time with each table requiring 32 bytes of information. Therefore, the total area allotted for color-table information is 512 bytes (16 tables × 32 bytes). Apple decided to place the color-table information after the location designated for scanline control byte information. They did not place color-table definitions *immediately* after the SCB information, but began them at memory location $9E00. Therefore, the destination address translated to decimal is 158 for the high-order byte ($9E) and 0 for the low-order byte (00).

Once the color table is defined and the correct addresses set, the color-table information can be transferred from bank 00 to bank 01 and shadowed into E1. The subroutine at line 17000 again accomplishes the transfer, and you are one step closer to placing information on the super high-resolution screen.

Clear the Super High-Resolution Screen

Before you can put information on the super high-resolution screen, you must first remove information from it. Once again, the process is not as simple as it sounds. The first problem is the size of the super high-resolution screen in comparison to the high-resolution screens. The high-resolution screens occupy 8K bytes of memory each for a total of 16K, and the super high-resolution screen takes up 32K bytes of memory (exactly 32000 bytes—160 bytes per line × 200 lines). *Information that is to be placed on the super high-resolution screen must originate within the high-resolution area of bank 00.* In other words, we cannot simply clear 32K bytes of memory in bank 00 and then transfer that cleared memory to bank 01 to be shadowed to bank E1. Instead, we must clear 16K bytes at a time. The process involves the following steps:

1. Clear High-Resolution Page 1
2. Clear High-Resolution Page 2

3. Transfer this 16K area of cleared memory to the first 16K area of super high-resolution memory
4. Reset the destination address to the start of the second half of super high-resolution memory
5. Transfer the same 16K area of cleared memory to the second 16K area of super high-resolution memory

The following BASIC instructions accomplish the task of clearing the super high-resolution screens:

```
835 GOSUB 15000:REM CLEAR SCREEN
8000 REM **--CLEAR HIRES MEMORY--**
8010 POKE 230,32:CALL-3086:REM CLEAR PAGE 1
8020 POKE 230,64:CALL-3086:REM CLEAR PAGE 2
8040 RETURN
8050:
8060:

15000 REM **--CLEAR SUPER RES SCREEN--**
15002:
15005 REM *-POKE ENDING ADDRESS-*
15010 EA = 24192:REM ENDING ADDRESS
15020 D = EA
15030 GOSUB 24000:REM CALCULATE LOW AND HIGH BYTE
15040 POKE 777,LB:REM POKE LOW BYTE
15050 POKE 781,HB:REM POKE HIGH BYTE
15052:
15055 REM *-POKE DESTINATION ADDRESS-*
15060 DA = 8192 :REM DESTINATION ADDRESS
15070 D = DA: GOSUB 24000:POKE 785,LB:POKE 789,HB
15080 GOSUB 8000:REM CLEAR HIRES SCREENS
15120 CALL MM:REM MOVE INFO TO E1 BANK
15130:
15200 REM *-POKE SECOND ENDING ADDRESS-*
15210 EA = 24192: D = EA:GOSUB 24000: POKE 777,LB:POKE 781,HB
15220:
15250 REM *-POKE SECOND DESTINATION ADDRESS-*
15260 EA = 24191:D = EA:GOSUB 24000:POKE 785,LB:POKE 789,HB
15310 GOSUB 8000:CALL MM
15330:
15500 REM *-RESET ROUTINE FOR ONE ADDRESS-*
15510 EA = 8192:D = EA:GOSUB 24000:POKE 777,LB:POKE 781,HB
15560 RETURN
15570:
15580:

24000 REM **--CALCULATE HIGH & LOW BYTE--**
```

```
24010 HB = INT (D/256)
24020 LB = D - 256 * HB
24030 RETURN
```

If you have read the previous chapters, the subroutine at line 8000 should be familiar. The subroutine at line 24000 takes a decimal address and converts it into its equivelent low-byte and high-byte values. These two values can then be POKEd into the appropriate locations in the assembly-language routine.

You are not actually using the entire 16K of high-resolution memory each time. The reason for this is that 16K equals 16,384 bytes of memory, not 16,000 bytes (16 × 1,024) (Remember, 1K, or kilobyte, equals 1024 bytes.) The ending address of 24192 produces exactly 16,000 bytes of memory (24192-8192) or exactly half of the super high-resolution memory. *Attempting to transfer the entire 16K high-resolution memory twice will erase the scanline control bytes and the color tables that occur immediately following the end of the super high-resolution screen memory.*

The reason for resetting the ending address as the last instruction in the subroutine at line 15000 will be explained a little later, but it is always a good idea to restore values or variables when a subroutine is finished using them.

PLOTTING DATA ON THE SUPER HIGH-RESOLUTION SCREEN

Finally, you are ready to access the super high-resolution screen and place some information on it. You will follow the same procedure you used to establish the scanline control bytes, create the color table, and clear the super high-resolution screen. Information is placed in a high-resolution memory location. That location is identified and passed to the assembly-language routine. The destination address is specified and also passed to the assembly-language routine. Finally the assembly-language routine AUXMOVE is called and a graphic display appears on the super high-resolution screen. The following BASIC code produces a small dot in the approximate center of the super high-resolution screen:

```
840 POKE 8192,15:REM SET COLOR OF POINT TO WHITE
845 D = 24112:GOSUB 24000:REM CALCULATE BYTE VALUE
850 POKE 785,LB:POKE 789,HB:REM SET NEW DESTINATION ADDRESS
855 CALL MM
860 INPUT "";NUL$
870 POKE 49193,1:END
880:
890:
```

If you have entered all the code accurately, a small white dot will appear in the approximate center of the super high-resolution screen when you run the program (Fig. 10.1). It will stay there until you press the RETURN key. At that point, the screen will return to the normal text mode.

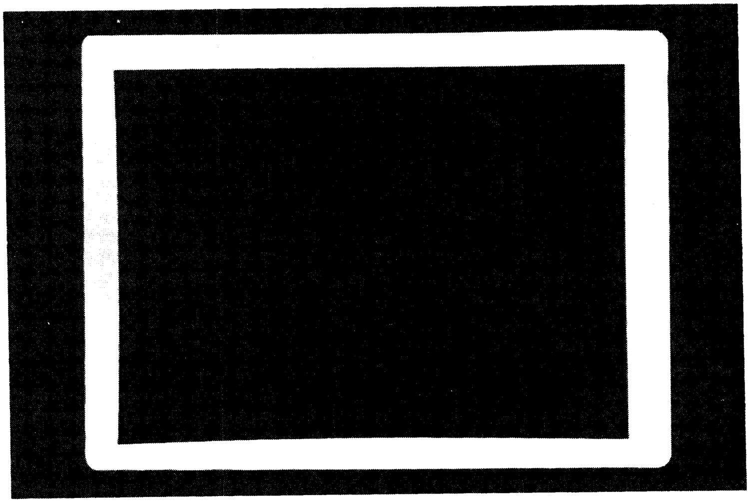


FIGURE 10.1. 320-mode super high-resolution point.

A small white dot may not seem like a great accomplishment, but when I started out, I was told that BASIC could not be used to access the super high-resolution screen at all. The procedure I have outlined to do it has gone through many refinements and appears to allow the BASIC programmer complete control over all the capabilities of the super high-resolution screen. Previous versions, especially of the assembly-language routine, restricted the ability to work with the entire super high-resolution screen, but this version is not restricted by the difference in the size between the high-resolution screens and the super high-resolution screen. You will see more of the flexibility of the routine a little later. The next task is to switch to 640 mode and plot a point.

640 Mode

A number of values can be used in the scanline control bytes to achieve 640-mode graphics, but for now use the decimal value 129 ($81). (Values from 0 to 127 define 320 mode, while values from 128 to 255 define 640 mode.) This value through its binary makeup (1000 0001) informs the computer that the mode should be 640 rather than 320 and that color table 1 (the second table we have defined) should be used. If you change the value in the SCB's from 0 to 129, the screen will switch from 320 mode to 640 mode. Theoretically, you should be able to make this one change and plot a point in 640 mode. Once again, things are more complicated than that.

The 640 mode handles colors in a different way than the 320 mode does. Since both modes must work with 160 bytes per line, 640 mode has only half the number of bits (two) in each point (pixel) as 320 mode has (four). Four bits means that each pixel in 320 mode

can indicate any of the 16 colors in a color table as its color. Two bits, however, means that each pixel in 640 mode can only indicate four of the 16 colors as its color. Therefore, 640 mode must handle colors differently. To accommodate this difference, you will create a separate color table for each mode. To do this, you need to add more DATA statements, which contain a 640-mode color-table definition, before you can see 640-mode graphics.

```
9500 REM **--640 MODE STANDARD COLOR TABLE--**
9510 DATA 0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520 DATA 0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9530:
9540:
```

Now you should be able to follow the procedure and plot a point on the super high-resolution screen in 640 mode. First, delete line 870 so that the program does not stop after displaying the white dot in 320 mode.

```
DEL 870
```

Then enter the following additional code:

```
900 REM **--640 MODE--**
905 M = 129:REM 640 MODE/COLOR TABLE #1
910 GOSUB 7000:REM POKE SCB'S
915 POKE 49193,163:REM SUPER RES
920 EH = 32 EL = 199:DH = 157:DL = 0:GOSUB 17000:REM MOVE SCB'S
925 GOSUB 7500:REM POKE 2ND COLOR TABLE
930 EH = 32:EL = 31:DH = 158:DL = 32:GOSUB 17000:REM MOVE COLOR
   TABLE
935 GOSUB 15000:REM CLEAR SCREEN
940 POKE 8192,1:REM SET COLOR OF POINT
945 D = 24112:GOSUB 24000:REM CALCULATE BYTE VALUE
950 POKE 785,LB:POKE 789,HB:REM SET NEW DEST.ADDRESS
955 CALL MM:REM MOVE INFO TO CORRECT LOCATION
960 INPUT "";NUL$
970 POKE 49193,1:END
980:
990:
```

When you run the program this time, you should still get the white dot in 320 mode, but after you press the RETURN key, the screen should clear and a very small blue dot should appear in approximately the same location as the white dot (Figs. 10.2 and 10.3). You have been able to plot a point on the super high-resolution screen in 640 mode, and you have a procedure that allows complete control over the full capabilities of super high-resolution graphics on the IIGS. Save the program at this point:

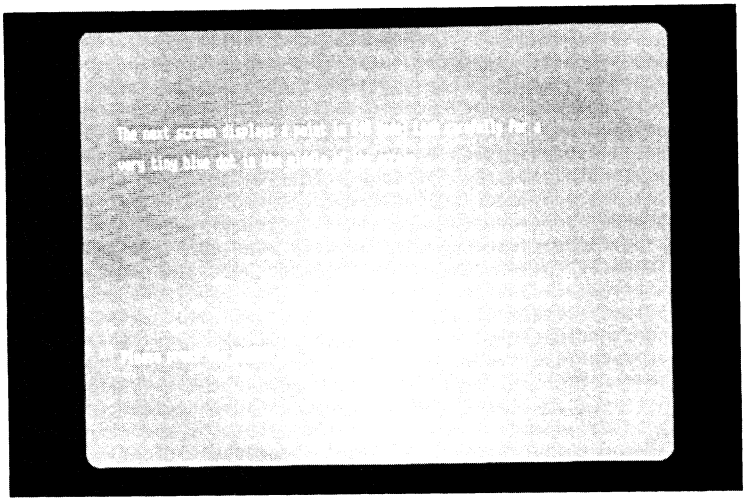


FIGURE 10.2. 640-mode single-point explanation.

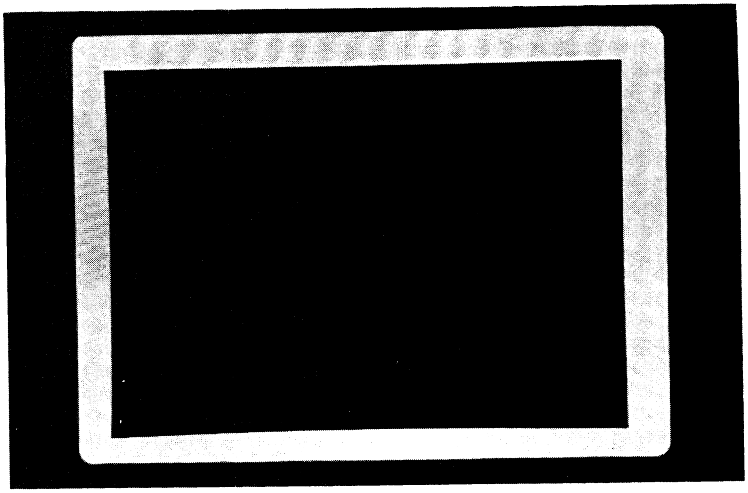


FIGURE 10.3. 640-mode single point (in center of screen).

```
SAVE SUPER.RES.DEMO1 {RETURN}
```

Now that the program is saved on diskette, you can make some changes and see more of how you can use BASIC to create super high-resolution graphics.

Horizontal Line

The next task is to create a horizontal line on the super high-resolution screen. Make the following changes and additions to the SUPER.RES.DEMO1 program:

```
842 FOR I = 0 TO 159
845 D = 24032 + I:GOSUB 24000:REM CALCULATE BYTE VALUE
857 NEXT I
942 FOR I = 0 TO 159
945 D = 24032 + I:GOSUB 24000:REM CALCULATE BYTE VALUE
957 NEXT I
```

Save this program as SUPER.RES.DEMO2 and then run it. You should first see a white horizontal line in the center of the screen (in 320 mode) (Figs. 10.4 and 10.5). After you press the RETURN key, you should see a blue horizontal line in 640 mode (Figs. 10.6 and 10.7).

The value of D contains the location on the screen where the computer plots the point. For each point, the computer converts the value of D into low-order and high-order decimal byte values, passes these values to the assembly-language routine as the new destination address and then calls the routine to move the contents of a single piece of information (the color code stored at 8192) and plots a point. This sequence can be repeated for *all* graphics work done on the super high-resolution screen. It is important, therefore, for you to completely understand the logic of this sequence and its flexibility. By providing the assembly-language routine with a new destination address for each point, you will be able to plot anywhere on the screen without being restricted by the smaller size of the high-resolution screen. You are not limited to plotting single points, either. If you want to move a block of data, you simply place that information in high-resolution memory (either

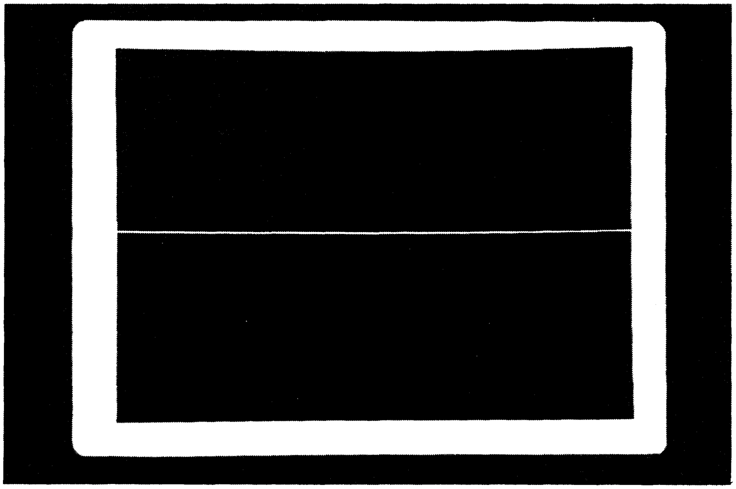


FIGURE 10.4. 320-mode horizontal line.

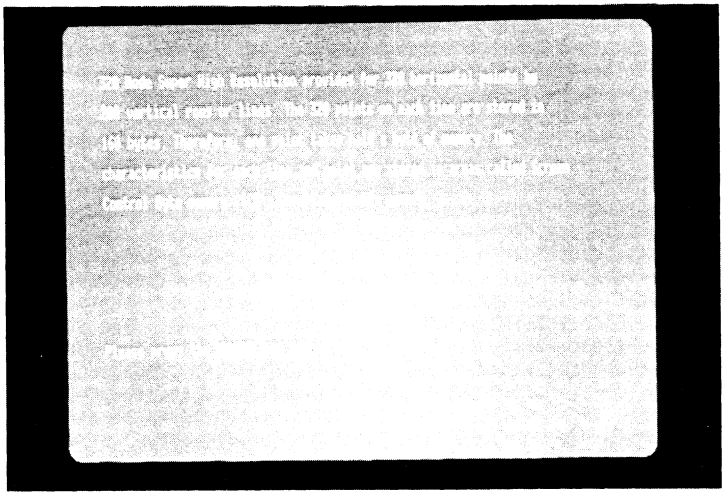


FIGURE 10.5. 320-mode explanation.

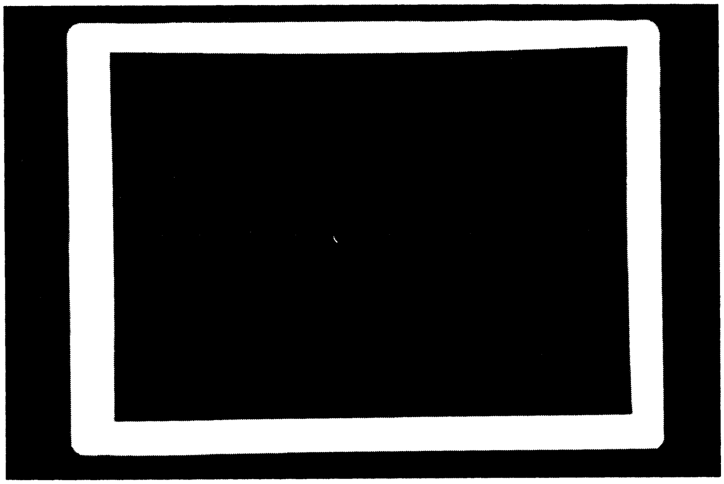


FIGURE 10.6. 640-mode horizontal line.

Page 1 or Page 2, or both), alter the ending address in the assembly-language routine and the destination address and again call on the routine to move the information from bank 00 to bank 01. Since bank 01 is shadowed into bank E1, the information will be placed exactly where you want it on the super high-resolution screen. The only real restriction is that you cannot move more than 16K of information at a time. If you want to recreate a super high-resolution screen that is full of information, it must be done in at least two steps,

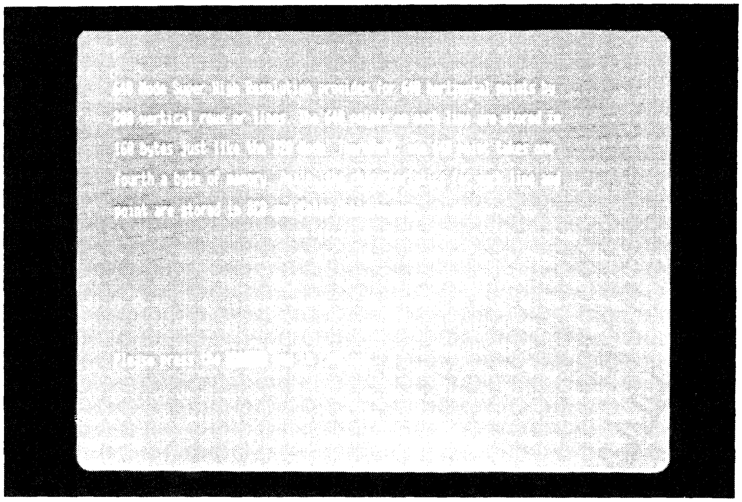


FIGURE 10.7. 640-mode explanation.

transferring no more than 16000 bytes of information at a time. (Review the information in the Clear Super High-Resolution Screen section of this chapter if you are unsure why you can transfer only 16000 bytes at a time.)

The value of the numeric variable D in lines 845 and 945 was changed from 24112 to 24032. To understand why I used the values 24112 and 24032, you must understand something about screen dimensions and screen memory.

SCREEN DIMENSIONS

The super high-resolution screen consists of 200 lines, each made up of 160 bytes of information. The location for the screen memory is in bank E1, address \$2000 to \$9CFF. Translated to decimal, the screen memory is at address 8192 to 40191. Unlike high-resolution screen memory, super high-resolution screen memory can be in a continuous form, the memory location of the first point in a new line occurs immediately after the memory location of the last point in the previous line. In other words, line nine comes after line 8 in memory just as it does on the screen. High-resolution memory does not follow this pattern and, as a result, has caused numerous programming problems. Therefore, if we know that the first point in the first line is memory location 8192 and that each line is 160 bytes long, we can calculate the location for any point on the screen. The first line starts at memory location 8192 and proceeds for a total of 160 bytes. The second line begins at memory location $8192 + 160$. Line 100 begins at memory location $8192 + ((100 - 1) \times 160)$ or 24032. The memory location for the middle of line 100 is $8192 + ((100 - 1) \times 160) + (160/2)$ or 24112. Using formulas like these, you can find the memory location of any

point on the screen, place a value in that location (color information) and plot that point. The only thing to remember is that the first point on the screen has a value of 8192 and that each line is 160 *bytes* (not points) long.

The fact that each line contains 160 bytes means that you can easily locate the byte location of each point—but that byte actually contains two points in 320 mode and four points in 640 mode. Placing a value in a byte will thus illuminate either two or four points depending upon the mode you are working in. You can still access a single point in any position by placing a value in the byte that contains only one part of the byte (half for 320 mode or one-fourth for 640 mode) that is different from the background. For example, the white dot was displayed by placing the value 15 in the proper byte location. In this case, that value of 15 was translated by the computer to mean that the left half of the byte contained a 0 while the right half contained an F (binary 1111). Therefore, only the right half of the byte displayed any information even though both points were actually plotted—one black point and one white point. Obviously, knowledge of your color table is important in producing desired results. If you want to plot two points side by side, you can choose a byte value that contains the two colors you want side by side or, if that is not possible in the table you are using, you can create another table that contains those two colors side by side. (I am not saying that the process is easy, simply that it is possible.) For example, in the current 320-mode color table, a byte value of 7 (decimal) produces a reddish-orange dot in the right-half portion of the byte. The decimal value 135 retains that reddish-orange right-half portion and adds a white dot in the left half of the byte. You can see this effect by modifying the SUPER.RES.DEMO1 program. Load the program and add the following lines:

```
865 N = VAL(NUL$):POKE 8192,N
867 IF NUL$ = "Q" THEN POKE 49193,1:END
868 GOTO 845
```

Now when you run the program, the same white dot will appear, but you can also enter a number between 0 and 255 and have that value plotted on the screen. The value 255 places two white dots side by side while the value 0 erases the dot completely. When you have finished trying different numbers, you can press the "Q" key, and the program will switch back to Text Page 1 and end. A somewhat more elaborate version of this modified program, DIFF.COLOR.MENU, is included at the end of the chapter.

At the end of this chapter I have included some programs that demonstrate the differences between the resolutions available on the IIGS. The color demonstration program takes a while to display all of the colors available, even with the IIGS operating in fast mode. Please read over these programs to see how the sequence presented in this chapter works. In the next chapter, I will discuss scanline control bytes in more detail and briefly examine the graphic ROM routines, TOOLBOX, included with every IIGS.

REVIEW QUESTIONS

1. Which bank of the IIGS's memory contains the super high-resolution memory?
2. Which memory location enables shadowing?
3. Which memory location switches the display to the super high-resolution screen?
4. The term SCB stands for what?
5. True or False. Super high-resolution has 16 colors available.
6. How many color tables can be available to super high-resolution programmers at one time?
7. What is the maximum number of colors possible in each color table?

Memory Mover

```
100  REM   ***--MOVE.MEMORY--***
110 :
120 :
130  REM  **--POKE ROUTINE--**
140  FOR L = 768 TO 796
150  READ V
160  POKE L,V
170  NEXT L
180 :
190 :
200  REM  **--END--**
210  END
220 :
230 :
500  REM  **--DATA VALUES--**
510 :
520  REM  *--STARTING ADDRESS--*
530  DATA  169,0
540  DATA  133,60
550  DATA  169,32
560  DATA  133,61
570 :
580  REM  *--ENDING ADDRESS--*
590  DATA  169,1
600  DATA  133,62
610  DATA  169,32
620  DATA  133,63
630 :
640  REM  *--DESTINATION ADDRESS--*
650  DATA  169,224
660  DATA  133,66
670  DATA  169,58
680  DATA  133,67
690 :
700  DATA  56: REM  SET CARRY BIT
710 :
720  DATA 32,17,195: REM  JUMP TO AUXMOVE ROUTINE AT C311
730 :
740  DATA  96: REM  RTS
```

Super High-Resolution Demo 1

```
100  REM  **--SUPER.RES.DEMO1--***
110 :
120 :
500  REM   **--Title--**
510  HOME : VTAB 5: HTAB 25
520  INVERSE : PRINT "320 & 640 Mode Single Point Comparison": NORMAL
530  VTAB 22: HTAB 28
540  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
550  INPUT " key: ";NUL$
560 :
570 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
798 :
799 :
800  REM   **--320 MODE SINGLE POINT--**
802 MESSAGE$ = "320 Mode Single Point"
803  GOSUB 3000: REM  Message routine
805 M = 0: REM  320 Mode
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color Tab
    le
835  GOSUB 15000: REM Clear Screen
840  POKE 8192,15: REM  Set color of point to white
845 D = 24112: GOSUB 24000: REM  Calculate Byte Value
850  POKE 785,LB: POKE 789,HB: REM  Set new destination address
855  CALL MM: REM Move Memory
860  INPUT " ";NUL$
870 :
880 :
900  REM  **--640 MODE SINGLE POINT--**
902 MESSAGE$ = "640 Mode Single Point"
903  GOSUB 3000: REM  Message routine
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM  Move Color
    Table
935  GOSUB 15000: REM Clear Screen
940  POKE 8192,1: REM   Set color of point
```

```
945 D = 24112: GOSUB 24000: REM   Calculate Byte Value
950  POKE 785,LB: POKE 789,HB: REM   Set new destination address
955  CALL MM: REM Move Info To Correct Location
960  INPUT " ";NUL$
980 :
990 :
2000  REM    **--End--**
2010  POKE 49193,1: REM   Switch to text
2020  HOME : VTAB 10
2030  PRINT "This has been a comparison between 320 Mode and 640 Mode Su
    per-res Graphics."
2040  END
2050 :
2060 :
3000  REM   **--Message On Text Page--**
3010  POKE 49193,1: REM   Return to text page
3020  HOME : VTAB 10
3030  PRINT MESSAGE$
3040  VTAB 22
3050  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
3060  INPUT " key: ";NUL$
3070  POKE 49193,163: REM  Super-res Screen
3080  RETURN
3090 :
3100 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
```

```
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM    **--640 Mode Standard Color Table--**
9510  DATA     0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA     0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000:  CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Super High-Resolution Demo 2

```
100 REM ***--SUPER.RES.DEMO2--***
110 :
120 :
500  REM  **--Title--**
510  HOME : VTAB 5: HTAB 25
520  INVERSE : PRINT "320 & 640 Mode Horiz. Line Comparison": NORMAL
530  VTAB 22: HTAB 28
540  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
550  INPUT " key: ";NUL$
560 :
570 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720  MM = 768: REM Move Memory--Location of Assembly Lang.routine
798 :
799 :
800  REM   **--320 MODE HORIZ.LINE--**
802 MESSAGE$ = "320 Mode Horizontal Line"
803  GOSUB 3000: REM Message routine
805 M = 0: REM 320 Mode
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color Tab
    le
835  GOSUB 15000: REM Clear Screen
840  POKE 8192,15: REM Set color
842  FOR I = 0 TO 159
845 D = 24032 + I: GOSUB 24000: REM Calculate Byte Value
850  POKE 785,LB: POKE 789,HB: REM Set new destination address
855  CALL MM: REM Move Memory
857  NEXT I
860  INPUT " ";NUL$
870 :
880 :
900  REM  **--640 MODE HORIZ.LINE--**
902 MESSAGE$ = "640 Mode Horizontal Line"
903  GOSUB 3000: REM  Message routine
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM  Move Color Table
935  GOSUB 15000: REM Clear Screen
940  POKE 8192,1: REM Set color
```

```
942  FOR I = 0 TO 159
945 D = 24032 + I: GOSUB 24000: REM Calculate Byte Value
950  POKE 785,LB: POKE 789,HB: REM Set new destination address
955  CALL MM: REM  Move Info To Correct Location
957  NEXT I
960  INPUT " ";NUL$
980 :
990 :
2000  REM   **--End--**
2010  POKE 49193,1: REM   Switch to text
2020  HOME : VTAB 10
2030  PRINT "This has been a comparison between 320 Mode and 640 Mode Su
    per-res Graphics."
2040  END
2050 :
2060 :
3000  REM   **--Message On Text Page--**
3010  POKE 49193,1: REM   Return to text page
3020  HOME : VTAB 10
3030  PRINT MESSAGE$
3040  VTAB 22
3050  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
3060  INPUT " key: ";NUL$
3070  POKE 49193,163: REM  Super-res Screen
3080  RETURN
3090 :
3100 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
```

```
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240 15,255,15
9540 :
9550 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Super High-Resolution Demo 3

```
100  REM    ***--SUPER.RES.DEMO3--***
110 :
120 :
500  REM   **--Title--**
510  HOME : VTAB 5: HTAB 26
520  INVERSE : PRINT "320 & 640 Mode Vert. Line Comparison": NORMAL
530  VTAB 22: HTAB 28
540  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
550  INPUT " key: ";NUL$
560 :
570 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
798 :
799 :
800  REM   **--320 MODE HORIZ.LINE--**
802 MESSAGE$ = "320 Mode Vertical Line"
803  GOSUB 3000: REM  Message routine
805 M = 0: REM  320 Mode
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color
    Table
835  GOSUB 15000: REM Clear Screen
840  POKE 8192,7: REM   Set color
842  FOR I = 8267 TO 40117 STEP 160
845 D = I: GOSUB 24000: REM Calculate Byte Value
850  POKE 785,LB: POKE 789,HB: REM Set new destination address
855  CALL MM: REM  Move Memory
857  NEXT I
860  INPUT " ";NUL$
870 :
880 :
900  REM   **--640 MODE HORIZ.LINE--**
902 MESSAGE$ = "640 Mode Vertical Line"
903  GOSUB 3000: REM Message routine
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
925  GOSUB 7500: REM Poke Color Table
```

```
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM Move Color Table

935  GOSUB 15000: REM Clear Screen
940  POKE 8192,1: REM  Set color
942  FOR I = 8267 TO 40117 STEP 160
945 D = I: GOSUB 24000: REM Calculate Byte Value
950  POKE 785,LB: POKE 789,HB: REM Set new destination address
955  CALL MM: REM  Move Info To Correct Location
957  NEXT I
960  INPUT " ";NUL$
980 :
990 :
2000  REM   **--End--**
2010  POKE 49193,1: REM   Switch to text
2020  HOME : VTAB 10
2030  PRINT "This has been a comparison between 320 Mode and 640 Mode Su
    per-res Graphics."
2040  END
2050 :
2060 :
3000  REM   **--Message On Text Page--**
3010  POKE 49193,1: REM   Return to text page
3020  HOME : VTAB 10
3030  PRINT MESSAGE$
3040  VTAB 22
3050  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
3060  INPUT " key: ":NUL$
3070  POKE 49193,163: REM  Super-res Screen
3080  RETURN
3090 :
3100 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
```

```
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
```

```
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Super High-Resolution Demo 4

```
100  REM   ***--SUPER.RES.DEMO4--***
110 :
120 :
500  REM   **-Title--**
510  HOME : VTAB 5: HTAB 29
520  INVERSE : PRINT "320 & 640 Color Comparison": NORMAL
530  VTAB 22: HTAB 28
540  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
550  INPUT " key: ";NUL$
560 :
570 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
798 :
799 :
800  REM   **--320 MODE SINGLE POINT--**
802 MESSAGE$ = "320 Mode Colors From Color Table #0. This Demo Takes a W
    hile."
803  GOSUB 3000: REM  Message routine
805 M = 0: REM  320 Mode
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color Tab
    le
835  GOSUB 15000: REM Clear Screen
840  POKE 8192,255 - K
842  FOR I = 0 TO 159
844 D = 8192 + L + I: GOSUB 24000: REM Calculate Byte Value
846  POKE 785,LB: POKE 789,HB: REM  Set new destination address
848  CALL MM: REM  Move Memory
850  NEXT I
852 K = K + 1:L = L + 160
854  IF K = 255 THEN 870
856  IF L + 8192 < 40031 THEN 840
858  GOSUB 15000: REM  Clear screen
860 L = 0
862  GOTO 840: REM  Finish colors
870  INPUT " ";NUL$
880 :
890 :
900  REM  **--640 MODE SINGLE POINT--**
902 MESSAGE$ = "640 Mode Colors From Color Table #1. This Demo Takes a W
    hile."
```

```
903  GOSUB 3000: REM  Message routine
904 K = 0:L = 0
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM  Move Color Table

935  GOSUB 15000: REM Clear Screen
940  POKE 8192,255 - K
942  FOR I = 0 TO 159
944 D = 8192 + L + I: GOSUB 24000: REM  Calculate Byte Value
946  POKE 785,LB: POKE 789,HB: REM   Set new destination address
948  CALL MM: REM   Move Memory
950  NEXT I
952 K = K + 1:L = L + 160
954  IF K = 255 THEN 970
956  IF L + 8192 < 40031 THEN 940
958  GOSUB 15000: REM   Clear screen
960 L = 0
962  GOTO 940: REM   Finish colors
970  INPUT " ";NUL$
980 :
990 :
2000  REM    **--End--**
2010  POKE 49193,1: REM   Switch to text
2020  HOME : VTAB 10
2030  PRINT "This has been a comparison between 320 Mode and 640 Mode Su
    per-res Graphics."
2040  END
2050 :
2060 :
3000  REM   **--Message On Text Page--**
3010  POKE 49193,1: REM   Return to text page
3020  HOME : VTAB 10
3030  PRINT MESSAGE$
3040  VTAB 22
3050  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
3060  INPUT " key: ";NUL$
3070  POKE 49193,163: REM  Super-res Screen
3080  RETURN
3090 :
3100 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
```

```
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
```

```
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Super High-Resolution Menu

```
100  REM  ***--SUPER HIGH RESOLUTION DEMONSTRATION MENU--***
110 :
120 :
130 TB = 25
140 D$ =  CHR$ (4): REM  CONTROL D
150 :
160 :
170  REM  **--DISPLAY MENU--**
180  TEXT : HOME
190  VTAB 2
200  HTAB TB + 4
210  INVERSE
220  PRINT "SUPER HIGH RESOLUTION MENU"
230  NORMAL
240  PRINT : PRINT
250  HTAB TB
260  PRINT "1.  Single Point Demonstration"
270  PRINT : HTAB TB
280  PRINT "2.  Horizontal Line Demonstration"
290  PRINT : HTAB TB
300  PRINT "3.  Vertical Line Demonstration"
310  PRINT : HTAB TB
320  PRINT "4.  Color Demonstration"
330  PRINT : HTAB TB
340  PRINT "5.  Circle Demonstration"
350  PRINT : HTAB TB
360  PRINT "6.  Diamond Demonstration"
370  PRINT : HTAB TB
380  PRINT "7.  List of Files on Diskette"
390  PRINT : HTAB TB
400  PRINT "8.  End This Demonstration"
410  PRINT : HTAB TB
420  PRINT "Which Number Please? (";: INVERSE : PRINT "1-8";: NORMAL : PRINT
    "):";
430  GET NB$: PRINT
440 NB =  VAL (NB$)
450  IF NB < 1 OR NB > 8 THEN PRINT : HTAB TB: INVERSE : PRINT "INCORRE
    CT CHOICE!": NORMAL : GOTO 190
460  IF NB = 1 THEN 1000
470  IF NB = 2 THEN 2000
480  IF NB = 3 THEN 3000
490  IF NB = 4 THEN 4000
500  IF NB = 5 THEN 5000
510  IF NB = 6 THEN 6000
520  IF NB = 7 THEN 7000
530  IF NB = 8 THEN 8000
```

```
540 :
550 :
1000  REM    **--SINGLE POINT--**
1010  PRINT D$;"RUN SINGLE.POINT"
1980 :
1990 :
2000  REM    **--HORIZONTAL LINE--**
2010  PRINT D$;"RUN HORIZ.LINE"
2980 :
2990 :
3000  REM    **-VERTICAL LINE--**
3010  PRINT D$;"RUN VERT.LINE"
3980 :
3990 :
4000  REM   **--COLORS--**
4010  PRINT D$;"RUN COLORS"
4980 :
4990 :
5000  REM   **--CIRCLE--**
5010  PRINT D$;"RUN CIRCLE"
5980 :
5990 :
6000  REM   **--RECTANGLE--**
6010  PRINT D$;"RUN DIAMOND"
6980 :
6990 :
7000  REM     **--DISPLAY CATALOG--**
7010  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
7020  HOME
7030  PRINT D$:"CATALOG"
7040  PRINT
7050  POKE  - 16368,0: REM   CLEAR KBRD BUFFER
7060  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " key to continue:";L$
7070  GOTO 170
7980 :
7990 :
8000  REM   **--END--**
8010  HOME
8020  VTAB 10
8030  INVERSE
8040  PRINT "See you next time."
8050  NORMAL
8060  END
9998 :
9999 :
10000  REM   SUPER-RES DEMO
10100  REM   COPYRIGHT JULY 1987
```

```
11000  REM   DAVID H. MILLER
13000  REM   1750 SULPHUR SPRINGS RD
14000  REM   CORVALLIS, OR 97330
15000  REM   503-745-7440
```

Single Point Demo

```
100  REM     ***--SINGLE.POINT--**
110 :
120 :
130  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
150  HOME : VTAB 3
155 TB = 20
160  HTAB 35: INVERSE : PRINT "SINGLE POINT": NORMAL
170  VTAB 20
210  HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue:";A$
215 TB = 10
220  HOME : VTAB 3: INVERSE : HTAB 35: PRINT "INSTRUCTIONS": NORMAL
230  VTAB 10: HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "RETU
    RN";: NORMAL : PRINT " key to switch between the various screens."
240  PRINT
260  PRINT : PRINT : PRINT
270  HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "ESC";: NORMAL
    : PRINT " key to return to the main menu."
280  PRINT : PRINT : HTAB 39: GET L$
290  IF  ASC (L$) = 27 THEN 5000
300  PRINT
310  IF  ASC (L$) <  > 13 THEN 230
320  HGR : GR
330  COLOR= 9
340  PLOT 20,20
350  HOME : VTAB 23
360  GOSUB 1000: REM  LOW-RES
370  IF  ASC (L$) = 27 THEN 5000
380  POKE  - 16297,0: REM  SWTICH DISPLAY TO HI-RES
390  HCOLOR= 6
400  HPLOT 142,75
460  VTAB 23: CALL  - 868: REM   CLEAR LINE
470  GOSUB 2000: REM  HI-RES
498 :
499 :
500  REM  **--Super Res Information--**
510  GOSUB 6000: REM  Text screen
```

```
515  PRINT "Next is a display of a single point on the Super High Resolu
    tion"
520  PRINT
525  PRINT "screen in 320 Mode. The screen operates much the same as doe
    s HGR2 in that"
530  PRINT
535  PRINT "there is not a direct way of displaying text on the screen.
    Therefore, "
540  PRINT
545  PRINT "you MUST remember to press the 'RETURN' key when you are rea
    dy to go on to"
550  PRINT
555  PRINT "the next display."
560  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
598 :
599 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
798 :
799 :
800  REM   **--320 MODE SINGLE POINT--**
805 M = 0: REM  320 Mode
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color Tab
    le
835  GOSUB 15000: REM Clear Screen
840 D = 24112: GOSUB 24000: REM Calculate Byte Value
845  POKE 785,LB: POKE 789,HB: REM Set new destination address
850  POKE 8192,15: REM Set color of point to white
855  CALL MM: REM Move Memory
860  INPUT A$: GOSUB 6000: REM  Supply text info
865  GOSUB 3000
870  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
898 :
899 :
900  REM  **--640 MODE SINGLE POINT--**
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM  Move Color Table
```

```
935  GOSUB 15000: REM Clear Screen
940 D = 24112: GOSUB 24000: REM  Calculate Byte Value
945  POKE 785,LB: POKE 789,HB: REM  Set new destination address
950  POKE 8192,1: REM  Set color of point
955  CALL MM: REM Move Info To Correct Location
960  INPUT A$: GOSUB 6000: REM  Supply text info
965  GOSUB 4000
970  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
975  GOTO 5000
998 :
999 :
1000  REM  **--LOW RESOLUTION--**
1010  INVERSE : PRINT "Low";: NORMAL : PRINT "Resolution--One Point:";:
     GET L$
1020  PRINT
1030  RETURN
1980 :
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  INVERSE : PRINT "High";: NORMAL : PRINT " Resolution--One Point:";
    : GET L$
2020  PRINT
2030  RETURN
2980 :
2990 :
3000  REM  **--320 Mode Information--**
3010  PRINT "320 Mode Super High Resolution provides for 320 horizontal
    points by "
3020  PRINT
3030  PRINT "200 vertical rows or lines. The 320 points on each line are
     stored in "
3040  PRINT
3050  PRINT "160 bytes. Therefore, one point takes half a byte of memory
    . The"
3060  PRINT
3070  PRINT "characteristics for each line and point are stored in areas
     called Scanline"
3080  PRINT
3090  PRINT "Control Byte memory (SCB's) and Color Table memory."
3100  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
3110  HOME : VTAB 5
3120  PRINT "The next screen displays a point in 640 Mode! Look carefull
    y for a"
3130  PRINT
3140  PRINT "very tiny blue dot in the middle of the screen."
```

```
3150  RETURN
3998 :
3999 :
4000  REM  **--640 Mode Information--**
4010  PRINT "640 Mode Super High Resolution provides for 640 horizontal
    points by"
4020  PRINT
4030  PRINT "200 vertical rows or lines. The 640 points on each line are
    stored in"
4040  PRINT
4050  PRINT "160 bytes just like the 320 Mode. Therefore, one 640 point
    takes one "
4060  PRINT
4070  PRINT "fourth a byte of memory. Again, the characteristics for eac
    h line and"
4080  PRINT
4090  PRINT  "point are stored in SCB and Color Table memory."
4100  RETURN
4998 :
4999 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 5
5020  PRINT "This program has provided a comparison between a low resolu
    tion point,"
5030  PRINT
5040  PRINT "a high resolution point, a super high resolution point in 3
    20 Mode, and"
5050  PRINT
5060  PRINT "a super high resolution point in 640 Mode."
5110  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
5120  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU. ";L$
5130  PRINT D$;"RUN SUPER.MENU"
5990 :
5995 :
6000  REM   **--RETURN TO THE TEXT SCREEN--**
6010  POKE 49193,1: REM  Return to normal display
6015  TEXT
6020  HOME
6030  VTAB 2
6040  RETURN
6050 :
6060 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
```

```
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM   **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192: D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
```

```
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EL: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Horizontal-Line Demo

```
100  REM  ***--HORIZ.LINE--***
110 :
120 :
130  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
150  HOME : VTAB 3
155 TB = 20
160  HTAB 35: INVERSE : PRINT "HORIZONTAL LINE": NORMAL
170  VTAB 20
210  HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
   : INPUT " key to continue:";A$
215 TB = 10
220  HOME : VTAB 3: INVERSE : HTAB 35: PRINT "INSTRUCTIONS": NORMAL
230  VTAB 10: HTAB TB: PRINT "Please press the ";: INVERSE : PRINTER "RETU
   RN";: NORMAL : PRINT " key to switch between the various screens."
240  PRINT
260  PRINT : PRINT : PRINT
270  HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "ESC";: NORMAL
   : PRINT " key to return to the main menu."
280  PRINT : PRINT : HTAB 39: GET L$
290  IF  ASC (L$) = 27 THEN 5000
300  PRINT
310  IF  ASC (L$) <  > 13 THEN 230
320  HGR : GR : COLOR= 1
330  HLIN 0,39 AT 20
340  HOME : VTAB 23
```

```
350  GOSUB 1000: REM LOW-RES
360  IF L$ = "S" OR L$ = "s" THEN 5000
370  POKE  - 16297,0: REM SWITCH DISPLAY TO HI-RES
380  HCOLOR= 6
390  HPLOT 0,80 TO 279,80
400  VTAB 23: CALL  - 878: REM CLEAR LINE
410  GOSUB 2000: REM HI-RES
498 :
499 :
500  REM  **--Super Res Information--**
510  GOSUB 6000: REM  Text screen
515  PRINT "Next is a display of a horiz. line on the Super High Resolut
    ion"
520  PRINT
525  PRINT "screen in 320 Mode. The screen operates much the same as doe
    s HGR2 in that"
530  PRINT
535  PRINT "there is not a direct way of displaying text on the screen.
    Therefore, "
540  PRINT
545  PRINT "you MUST remember to press the 'RETURN' key when you are rea
    dy to go on to"
550  PRINT
555  PRINT "the next display."
560  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
598 :
599 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
798 :
799 :
800  REM **--320 MODE HORIZONTAL LINE--**
805 M = 0: REM  320 Mode
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color Tab
    le
835  GOSUB 15000: REM Clear Screen
840 C = 138: REM  Color of line
845  GOSUB 12000: REM  Horizontal line routine
860  INPUT A$: GOSUB 6000: REM  Supply text info
865  GOSUB 3000
870  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
```

```
898 :
899 :
900  REM   **--640 MODE HORIZ. LINE--**
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM Move Color Table
935  GOSUB 15000: REM Clear Screen
940 C = 4: REM Color of line in 640 Mode
945  GOSUB 12000: REM  Horizontal line routine
960  INPUT A$: GOSUB 6000: REM  Supply text info
965  GOSUB 4000: REM  640 INFO
970  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
975  GOTO 5000: REM  RETURN TO MENU
998 :
999 :
1000  REM  **--LOW RESOLUTION--**
1010  INVERSE : PRINT "Low";: NORMAL : PRINT " Resolution--Horizontal li
    ne (40 points):";: GET L$
1020  PRINT
1030  RETURN
1980 :
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  INVERSE : PRINT "High";: NORMAL : PRINT " Resolution--Horizontal l
    ine (280 points):";: GET L$
2020  PRINT
2030  RETURN
2980 :
2990 :
3000  REM **--320 Mode Information--**
3010  PRINT "320 Mode Super High Resolution provides for 320 horizontal
    points by "
3020  PRINT
3030  PRINT "200 vertical rows or lines. The 320 points on each line are
     stored in "
3040  PRINT
3050  PRINT "160 bytes. Therefore, one point takes half a byte of memory
    . The"
3060  PRINT
3070  PRINT "characteristics for each line and point are stored in areas
     called Scanline"
3080  PRINT
3090  PRINT "Control Byte memory (SCB's) and Color Table memory."
```

```
3100  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
3110  HOME : VTAB 5
3120  PRINT "The next screen displays a line in 640 Mode!"
3150  RETURN
3998 :
3999 :
4000  REM  **--640 Mode Information--**
4010  PRINT "640 Mode Super High Resolution provides for 640 horizontal
    points by"
4020  PRINT
4030  PRINT "200 vertical rows or lines. The 640 points on each line are
     stored in"
4040  PRINT
4050  PRINT "160 bytes just like the 320 Mode. Therefore, one 640 point
    takes one "
4060  PRINT
4070  PRINT "fourth a byte of memory. Again, the characteristics for eac
    h line and"
4080  PRINT
4090  PRINT "point are stored in SCB and Color Table memory."
4100  RETURN
4998 :
4999 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 5
5020  PRINT "This program has provided a comparison between a low resolu
    tion line,"
5030  PRINT
5040  PRINT "a high resolution line, a super high resolution line in 320
     Mode, and"
5050  PRINT
5060  PRINT "a super high resolution line in 640 Mode."
5110  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
5120  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU. ";L$
5130  PRINT D$;"RUN SUPER.MENU"
5990 :
5995 :
6000  REM   **--RETURN TO THE TEXT SCREEN--**
6010  POKE 49193,1: REM  Return to normal display
6015  TEXT
6020  HOME
6030  VTAB 2
6040  RETURN
6050 :
6060 :
7000  REM  **--Scanline Control Bytes--**
```

```
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
12000  REM **--HORIZONTAL LINE--**
12025  POKE 8192,C
12030  FOR I = 0 TO 159
12035 D = 24032 + I
12040  GOSUB 24000: POKE 785,LB: POKE 789,HB
12045  CALL MM
12050  NEXT I
12055  RETURN
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
```

```
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Vertical-Line Demo

```
100  REM  ***--VERT.LINE--***
110 :
120 :
130  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
150  HOME : VTAB 3
155 TB = 20
160  HTAB 35: INVERSE : PRINT "VERTICAL LINE": NORMAL
170  VTAB 20
210  HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue:";A$
215 TB = 10
220  HOME : VTAB 3: INVERSE : HTAB 35: PRINT "INSTRUCTIONS": NORMAL
```

```
230  VTAB 10: HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "RETU
    RN";: NORMAL : PRINT " key to switch between the various screens."
240  PRINT
260  PRINT : PRINT : PRINT
270  HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "ESC";: NORMAL
    : PRINT " key to return to the main menu."
280  PRINT : PRINT : HTAB 39: GET L$
290  IF  ASC (L$) = 27 THEN 5000
300  PRINT
310  IF  ASC (L$) <  > 13 THEN 230
320  HGR : GR
330  COLOR= 13
340  VLIN 0,39 AT 20
350  HOME : VTAB 23
360  GOSUB 1000: REM LOW-RES INFO
370  IF  ASC (L$) = 27 THEN 5000
380  POKE  - 16297,0: REM SWITCH DISPLAY TO HI-RES
390  HCOLOR= 1
400  HPLOT 139,0 TO 139,159
410  VTAB 23: CALL  - 868: REM CLEAR LINE
420  GOSUB 2000: REM HI-RES INFO
425  IF  ASC (L$) = 27 THEN 5000
498 :
499 :
500  REM  **--Super Res Information--**
510  GOSUB 6000: REM  Text screen
515  PRINT "Next is a display of a vert. line on the Super High Resoluti
    on"
520  PRINT
525  PRINT "screen in 320 Mode. The screen operates much the same as doe
    s HGR2 in that"
530  PRINT
535  PRINT "there is not a direct way of displaying text on the screen.
    Therefore, "
540  PRINT
545  PRINT "you MUST remember to press the 'RETURN' key when you are rea
    dy to go on to"
550  PRINT
555  PRINT "the next display."
560  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
598 :
599 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
798 :
799 :
```

```
800  REM  **--320 MODE VERTICAL LINE--**
805 M = 0: REM  320 Mode
810  GOSUB 7000: REM  Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199: DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color Tab
    le
835  GOSUB 15000: REM Clear Screen
840  POKE 8192,165
845  FOR I = 8267 TO 40117 STEP 160
850 D = I: GOSUB 24000: REM  Calculate Byte Value
855  POKE 785,LB: POKE 789,HB: REM  Set new destination address
860  CALL MM: REM  Move Memory
865  NEXT I
885  INPUT A$: GOSUB 6000: REM Supply text info
890  GOSUB 3000
895  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
898 :
899 :
900  REM  **--640 MODE VERTICAL LINE--**
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM Move Color Table
935  GOSUB 15000: REM Clear Screen
940  POKE 8192,1: REM  Set color of line
945  FOR I = 8267 TO 40117 STEP 160
950 D = I: GOSUB 24000: REM  Calculate Byte Value
955  POKE 785,LB: POKE 789,HB: REM  Set new destination address
960  CALL MM: REM  Move Memory
965  NEXT I
985  INPUT A$: GOSUB 6000: REM Supply text info
990  GOSUB 4000
995  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
997  GOTO 5000
998 :
999 :
1000  REM  **--LOW RESOLUTION--**
1010  INVERSE : PRINT "Low";: NORMAL : PRINT " Resolution--Vertical line
     (40 points):";: GET L$
1020  PRINT
1030  RETURN
1980 :
```

```
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  INVERSE : PRINT "High";: NORMAL : PRIINT " Resolution--Vertical lin
    e (160 points):";: GET L$
2020  PRINT
2030  RETURN
2980 :
2990 :
3000  REM  **--320 Mode Information--**
3010  PRINT "320 Mode Super High Resolution provides for 320 horizontal
    points by "
3020  PRINT
3030  PRINT "200 vertical rows or lines. The 320 points on each line are
     stored in "
3040  PRINT
3050  PRINT "160 bytes. Therefore, one point takes half a byte of memory
    . The"
3060  PRINT
3070  PRINT "characteristics for each line and point are stored in areas
     called Scanline"
3080  PRINT
3090  PRINT "Control Byte memory (SCB's) and Color Table memory."
3100  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
3110  HOME : VTAB 5
3120  PRINT "The next screen displays a line in 640 Mode!"
3150  RETURN
3998 :
3999 :
4000  REM  **--640 Mode Information--**
4010  PRINT "640 Mode Super High Resolution provides for 640 horizontal
    points by"
4020  PRINT
4030  PRINT "200 vertical rows or lines. The 640 points on each line are
    stored in"
4040  PRINT
4050  PRINT "160 bytes just like the 320 Mode. Therefore, one 640 point
    takes one "
4060  PRINT
4070  PRINT "fourth a byte of memory. Again, the characteristics for eac
    h line and"
4080  PRINT
4090  PRINT "point are stored in SCB and Color Table memory."
4100  RETURN
4998 :
4999 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 5
```

```
5020  PRINT "This program has provided a comparison between a low resolu
    tion line,"
5030  PRINT
5040  PRINT "a high resolution line, a super high resolution line in 320
     Mode, and"
5050  PRINT
5060  PRINT "a super high resolution line in 640 Mode."
5110  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
5120  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU.  ";L$
5130  PRINT D$;"RUN SUPER.MENU"
5990 :
5995 :
6000  REM  **--RETURN TO THE TEXT SCREEN--**
6010  POKE 49193,1: REM  Return to normal display
6015  TEXT
6020  HOME
6030  VTAB 2
6040  RETURN
6050 :
6060 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
```

```
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
12000  REM **--HORIZONTAL LINE--**
12025  POKE 8192,C
12030  FOR I = 0 TO 159
12035 D = 24032 + I
12040  GOSUB 24000: POKE 785,LB: POKE 789,HB
12045  CALL MM
12050  NEXT I
12055  RETURN
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
```

```
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Color Demo

```
100  REM  ***--COLORS--***
110 :
120 :
500  REM   **--Title--**
510  HOME : VTAB 5: HTAB 29
520  INVERSE : PRINT "320 & 640 Color Comparison": NORMAL
530  VTAB 22: HTAB 28
540  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
550  INPUT " key: ";NUL$
560 :
570 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
730 D$ =  CHR$ (4): REM  CONTROL D
798 :
799 :
800  REM   **--320 MODE SINGLE POINT--**
802 MESSAGE$ = "320 Mode Colors From Color Table #0. This Demo Takes a W
    hile."
803  GOSUB 3000: REM  Message routine
805 M = 0: REM  320 Mode
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color Tab
    le
835  GOSUB 15000: REM Clear Screen
840  POKE 8192,255 - K
842  FOR I = 0 TO 159
844 D = 8192 + L + I: GOSUB 24000: REM Calculate Byte Value
846  POKE 785,LB: POKE 789,HB: REM  Set new destination address
848  CALL MM: REM  Move Memory
850  NEXT I
852 K = K + 1:L = L + 160
854  IF K = 255 THEN 870
856  IF L + 8192 < 40031 THEN 840
858  GOSUB 15000: REM  Clear screen
860 L = 0
```

```
862  GOTO 840: REM  Finish colors
870  INPUT " ";NUL$
880 :
890 :
900  REM  **--640 MODE SINGLE POINT--**
902 MESSAGE$ = "640 Mode Colors From Color Table #1. This Demo Takes a W
    hile."
903  GOSUB 3000: REM  Message routine
904 K = 0:L = 0
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM  Move Color Table
935  GOSUB 15000: REM Clear Screen
940  POKE 8192,255 - K
942  FOR I = 0 TO 159
944 D = 8192 + L + I: GOSUB 24000: REM  Calculate Byte Value
946  POKE 785,LB: POKE 789,HB: REM   Set new destination address
948  CALL MM: REM   Move Memory
950  NEXT I
952 K = K + 1:L = L + 160
954  IF K = 255 THEN 970
956  IF L + 8192 < 40031 THEN 940
958  GOSUB 15000: REM   Clear screen
960 L = 0
962  GOTO 940: REM   Finish colors
970  INPUT " ";NUL$
980 :
990 :
2000  REM   **--End--**
2010  POKE 49193,1: REM   Switch to text
2020  HOME : VTAB 10
2030  PRINT "This has been a comparison between 320 Mode and 640 Mode Su
    per-res Graphics."
2040  VTAB 20: PRINT "Press the ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " key for the menu. ";L$
2045  PRINT D$;"RUN SUPER.MENU"
2050 :
2060 :
3000  REM   **--Message On Text Page--**
3010  POKE 49193,1: REM   Return to text page
3020  HOME : VTAB 10
3030  PRINT MESSAGE$
3040  VTAB 22
```

```
3050  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
3060  INPUT " key: ";NUL$
3070  POKE 49193,163: REM  Super-res Screen
3080  RETURN
3090 :
3100 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,255,15
9540 :
9550 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
```

```
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Circle Demo

```
100  REM  ***--CIRCLE--***
110 :
120 :
130  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
150  HOME : VTAB 3
155 TB = 20
160  HTAB 35: INVERSE : PRINT "CIRCLE SHAPE": NORMAL
170  VTAB 20
210  HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue:";A$
215 TB = 10
220  HOME : VTAB 3: INVERSE : HTAB 35: PRINT "INSTRUCTIONS": NORMAL
```

```
230  VTAB 10: HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "RETU
    RN";: NORMAL : PRINT " key to switch between the various screens."
240  PRINT
260  PRINT : PRINT : PRINT
270  HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "ESC";: NORMAL
    : PRINT " key to return to the main menu."
280  PRINT : PRINT : HTAB 39: GET L$
290  IF  ASC (L$) = 27 THEN 5000
300  PRINT
310  IF  ASC (L$) <  > 13 THEN 230
320  HGR : GR
330  HGR : GR
340  COLOR= 13
350 X1 = 20:Y1 = 20:P = .75:R = 11
360  FOR K = 0 TO 6.4 STEP .05
370  GOSUB 4500: REM  CIRCLE ROUTINE
380  PLOT X,Y
390  NEXT K
400  VTAB 23
410  GOSUB 1000: REM LOW-RES INFO
420  IF L$ = "S" OR L$ = "s" THEN 5000
425  IF  ASC (L$) = 27 THEN 5000
430  HOME
440  POKE  - 16297,0: REM    SWITCH DISPLAY TO HI-RES
450  HCOLOR= 3
460 X1 = 140:Y1 = 80:P = 1.16:4 = 45
470  FOR K = 0 TO 6.4 STEP .05
480  GOSUB 4500: REM      CIRCLE ROUTINE
490  IF K = 0 THEN  HPLOT X,Y: GOTO 510
500  HPLOT  TO X,Y
510  NEXT K
520  VTAB 23: CALL  - 868: REM    CLEAR LINE
530  GOSUB 2000: REM    HI-RES INFO
570 :
580 :
598 :
599 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
798 :
799 :
800  REM  **--320 MODE CIRCLE--**
801 M = 0: REM   320 Mode
802  GOSUB 7000: REM  Poke SCB's
803  POKE 49193,163: REM  Super Res
804 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM  Move SCB's
806  GOSUB 7500: REM  Poke Color Table
```

```
807 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM  Move 320 Color Ta
    ble
808  GOSUB 15000: REM  Clear Screen
809 PI = 3.14159
810 A1 = 0:A2 = 2 * PI:N = 512:R = 80
820 INC = (A2 - A1) / N
825  POKE 8192,64
830  FOR I = A1 TO A2 STEP INC
835 X = R *  SIN (I)
840 Y = R *  COS (I)
845 Y = Y + 50:X = X + 40
850 Y =  INT (Y)
855 Y = Y * 160
860 BP = 15288
865 X = X * .66
870 D = BP + Y + X
875  GOSUB 24000
880 LB = D - 256 * HB
882  POKE 785,LB
883  POKE 789,HB
887  CALL MM
888  NEXT I
889  INPUT A$: GOSUB 6000: REM  Supply text info
890  GOSUB 3000
895  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
898 :
899 :
900  REM  **--640 MODE CIRCLE--**
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM  Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM Move Color Table
935  GOSUB 15000: REM Clear Screen
940 PI = 3.14159
942 A1 = 0:A2 = 2 * PI:N = 512:R = 80
944 INC = (A2 - A1) / N
945  POKE 8192,4
946  FOR I = A1 TO A2 STEP INC
948 X = R *  SIN (I)
950 Y = R *  COS (I)
952 Y = Y + 50:X = X + 40
954 Y =  INT (Y)
955 Y = Y * 160
956 BP = 15288
958 X = X * .66
```

```
960 D = BP + Y + X
962  GOSUB 24000
964 LB = D - 256 * HB
965  POKE 785,LB
966  POKE 789,HB
970  CALL MM
972  NEXT I
985  INPUT A$: GOSUB 6000: REM Supply text info
990  GOSUB 4000
995  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
997  GOTO 5000
998 :
999 :
1000  REM  **--LOW RESOLUTION--**
1010  INVERSE : PRINT "Low";: NORMAL : PRINT " Resolution--Circle:";: GET
    L$
1020  PRINT
1030  RETURN
1980 :
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  INVERSE : PRINT "High";: NORMAL : PRINT " Resolution--Circle:";: GET
    L$
2020  PRINT
2030  RETURN
2980 :
2990 :
3000  REM  **--320 Mode Information--**
3010  PRINT "320 Mode Super High Resolution provides for 320 horizontal
    points by "
3020  PRINT
3030  PRINT "200 vertical rows or lines. The 320 points on each line are
     stored in "
3040  PRINT
3050  PRINT "160 bytes. Therefore, one point takes half a byte of memory
    . The"
3060  PRINT
3070  PRINT "characteristics for each line and point are stored in areas
     called Scanline"
3080  PRINT
3090  PRINT "Control Byte memory (SCB's) and Color Table memory."
3100  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
3110  HOME : VTAB 5
3120  PRINT "The next screen displays a circle in 640 Mode!"
3150  RETURN
3998 :
```

```
3999 :
4000  REM  **--640 Mode Information--**
4010  PRINT "640 Mode Super High Resolution provides for 640 horizontal
    points by"
4020  PRINT
4030  PRINT "200 vertical rows or lines. The 640 points on each line are
     stored in"
4040  PRINT
4050  PRINT "160 bytes just like the 320 Mode. Therefore, one 640 point
    takes one "
4060  PRINT
4070  PRINT "fourth a byte of memory. Again, the characteristics for eac
    h line and"
4080  PRINT
4090  PRINT "point are stored in SCB and Color Table memory."
4100  RETURN
4500  REM     **--CIRCLE ROUTINE--**
4510 X2 = R *  COS (K)
4520 Y2 = R *  SIN (K)
4530 X = P * X2 + X1
4540 Y = Y1 - Y2
4550  RETURN
4998 :
4999 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 5
5020  PRINT "This program has provided a comparison between a low resolu
    tion circle,"
5030  PRINT
5040  PRINT "a high resolution circle, a super high resolution circle in
     320 Mode, and"
5050  PRINT
5060  PRINT "a super high resolution circle in 640 Mode."
5110  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
5120  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU. ";L$
5130  PRINT D$;"RUN SUPER.MENU"
5990 :
5995 :
6000  REM  **--RETURN TO THE TEXT SCREEN--**
6010  POKE 49193,1:REM  Return to normal display
6015  TEXT
6020  HOME
6030  VTAB 2
6040  RETURN
6050 :
6060 :
7000  REM  **--Scanline Control Bytes--**
```

```
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
```

```
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Diamond Demo

```
100  REM    ***--DIAMOND--***
110 :
120 :
500  REM   **--Title--**
510  HOME : VTAB 5: HTAB 35
520  INVERSE : PRINT "Diamond Shape ": NORMAL
530  VTAB 22: HTAB 28
540  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
550  INPUT " key: ";NUL$
560 :
570 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
730 D$ =  CHR$ (4): REM  Control D
798 :
799 :
800  REM    **--320 MODE--**
802 MESSAGE$ = "The next screen shows a 320 Mode diamond drawn with diff
    erent colors."
803  GOSUB 3000: REM  Message routine
805 M = 0: REM   320 MODE
```

```
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color Tab
    le
835  GOSUB 15000: REM Clear Screen
840  FOR I = 1 TO 159
842  IF I > 79 THEN 860
843  POKE 8192,102: REM  Set color
845 ROW = I:COL = 79 + I: GOSUB 22000:D = ML: GOSUB 24000
847  POKE 785,LB: POKE 789,HB: REM   Set new destination address
849  CALL MM: REM  Move Memory
851  POKE 8192,153: REM  Set color of point
853 ROW = I:COL = 79 - I: GOSUB 22000:D = ML: GOSUB 24000
855  POKE 785,LB: POKE 789,HB: REM    Set new destination address
857  CALL MM: REM   Move Memory
859  GOTO 880
860  POKE 8192,112: REM   Set color of point
862 ROW = I:COL = 160 - (I - 79): GOSUB 22000:D = ML: GOSUB 24000
864  POKE 785,LB: POKE 789,HB: REM   Set new destination address
866  CALL MM: REM  Move Memory
868  POKE 8192,5: REM   Set color of point
870 ROW = I:COL = I - 79: GOSUB 22000:D = ML: GOSUB 24000
872  POKE 785,LB: POKE 789,HB: REM    Set new destination address
874  CALL MM: REM   Move Memory
880  NEXT I
890  INPUT " ";NUL$
898 :
899 :
900  REM   **--640 MODE--**
902 MESSAGE$ = "The next screen shows a 640 Mode diamond drawn with diff
    erent colors."
903  GOSUB 3000: REM  Message routine
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM Move Color Table
935  GOSUB 15000: REM Clear Screen
940  FOR I = 1 TO 159
942  IF I > 79 THEN 960
943  POKE 8192,4: REM  Set color of point
945 ROW - I:COL = 79 + I: GOSUB 22000:D = ML: GOSUB 24000
947  POKE 785,LB: POKE 789,HB: REM  Set new destination address
949  CALL MM: REM  Move Memory
951  POKE 8192,5: REM Set color of point
```

```
953 ROW = I:COL = 79 - I: GOSUB 22000:D = ML: GOSUB 24000
955  POKE 785,LB: POKE 789,HB: REM  Set new destination address
957  CALL MM: REM  Move Memory
959  GOTO 980
960  POKE 8192,6: REM Set color of point
962 ROW = I:COL = 160 - (I - 79): GOSUB 22000:D = ML: GOSUB 24000
964  POKE 785,LB: POKE 789,HB: REM  Set new destination address
966  CALL MM: REM Move Memory
968  POKE 8192,7: REM Set color of point
970 ROW = I:COL = I - 79: GOSUB 22000:D = ML: GOSUB 24000
972  POKE 785,LB: POKE 789,HB: REM  Set new destination address
974  CALL MM: REM  Move Memory
980  NEXT I
990  INPUT " ";NUL$
998 :
999 :
2000  REM    **--End--**
2010  POKE 49193,1: REM   Switch to text
2020  HOME : VTAB 10
2030  PRINT "This has been a demonstration of drawing w/diff.colors in d
    iff.directions."
2040  VTAB 20: PRINT "Press the ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " key for the menu. ";L$
2045  PRINT D$;"RUN SUPER.MENU"
2050 :
2060 :
3000  REM   **--Message On Text Page--**
3010  POKE 49193,1: REM   Return to text page
3020  HOME : VTAB 10
3030  PRINT MESSAGE$
3040  VTAB 22
3050  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
3060  INPUT " key: ";NUL$
3070  POKE 49193,163
3080  RETURN
3090 :
3100 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
```

```
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM    **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9030 :
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
```

```
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
22000  REM  **--Calculate Memory Location From Col.& Row Value--**
22010 ML = 8192 + ((ROW - 1) * 160) + COL
22020  RETURN
22030 :
22040 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Different Colors Menu

```
100  REM    ***--DIFF.COLOR.MENU--***
110 :
120 :
300  REM  **--Load Color Tables--**
310  HOME : VTAB 5: HTAB 30
320  PRINT "One Moment Please..."
325  POKE 49205,0: POKE 49193,163
327 MM = 768
330  GOSUB 7500: REM  Poke Color Table
340 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM  Move 320 Color Ta
    ble
350  GOSUB 7500: REM  Poke Color Table
360 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM  Move 320 Color T
    able
370 :
380 :
500  REM  **--Menu & Instructions--**
505  POKE 49193,1
510  HOME : VTAB 5: HTAB 30
520  INVERSE : PRINT "Colors Menu": NORMAL
530  VTAB 7: HTAB 30
540  PRINT "1.  320 Mode"
550  HTAB 30
560  PRINT "2.  640 Mode"
565  HTAB 30
567  PRINT "0. End Prog."
```

```
570  PRINT : PRINT
580  HTAB 30
590  INPUT "Which number please? ";N
595  IF N = 0 THEN  END
600  PRINT : PRINT
610  HOME : VTAB 10: HTAB 10
620  PRINT "Enter a number (0-255) for a color--the letter 'Q' to quit."
630  PRINT : PRINT
640  HTAB 30
650  INPUT "Press the RETURN key: ";NUL$
660 :
670 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
730  IF N = 1 THEN 800
740  IF N = 2 THEN 900
798 :
799 :
800  REM    **--320 MODE--**
805 M = 0: REM  320 Mode
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
835  GOSUB 15000: REM Clear Screen
840  POKE 8192,15: REM  Set color of point to white
841 Q = 24112
842  FOR J = 1 TO 5
844  FOR I = 1 TO 5
845 D = Q + I: GOSUB 24000: REM    Calculate Byte Value
850  POKE 785,LB: POKE 789,HB: REM  Set new destination address
855  CALL MM: REM Move Memory
857  NEXT I
858 Q = Q + 160
859  NEXT J
860  INPUT A$:
861  IF A$ = "Q" OR A$ = "q" THEN  POKE 49193,1: GOTO 500
862 A =  VAL (A$): POKE 8192,A: GOTO 841
898 :
899 :
900  REM   **--640 MODE--**
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
935  GOSUB 15000: REM Clear Screen
940  POKE 8192,15: REM   Set color of point to white
```

```
941 Q = 24112
942  FOR J = 1 TO 5
944  FOR I = 1 TO 5
945 D = Q + 1: GOSUB 24000: REM   Calculate Byte Value
950  POKE 785,LB: POKE 789,HB: REM   Set new destination address
955  CALL MM: REM  Move Memory
957  NEXT I
958 Q = Q + 160
959  NEXT J
960  INPUT A$;
961  IF A$ = "Q" OR A$ = "q" THEN  POKE 49193,1: GOTO 500
962 A =  VAL (A$): POKE 8192,A: GOTO 941
998 :
999 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8500 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
```

```
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310   GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D = 256 * HB
24030  RETURN
```

Special Circle Demo

```
100  REM     ***--SPECIAL.CIRCLE--***
110 :
120 :
130  POKE  - 16368,0: REM  CLEAR KBRD BUFFER
140 D$ =  CHR$ (4): REM  CONTROL D
145 MM = 768: POKE 49205,0: POKE 49193,163: GOSUB 15000: POKE 49193,1
150  HOME : VTAB 3
```

```
155 TB = 20
160  HTAB 35: INVERSE : PRINT "CIRCLE SHAPE": NORMAL
170  VTAB 20
210  HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
   : INPUT " key to continue:";A$
215 TB = 10
220  HOME : VTAB 3: INVERSE : HTAB 35: PRINT "INSTRUCTIONS": NORMAL
230  VTAB 10: HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "RETU
   RN";: NORMAL : PRINT " key to switch between the various screens."
240  PRINT
260  PRINT : PRINT : PRINT
270  HTAB TB: PRINT "Please press the ";: INVERSE : PRINT "ESC";: NORMAL
   : PRINT " key to return to the main menu."
280  PRINT : PRINT : HTAB 39: GET L$
290  IF  ASC (L$) = 27 THEN 5000
300  PRINT
310  IF  ASC (L$) <  > 13 THEN 230
320  HGR : GR
330  HGR : GR
340  COLOR= 13
350 X1 = 20:Y1 = 20:P = .75:R = 11
360  FOR K = 0 TO 6.4 STEP .05
370  GOSUB 4500: REM  CIRCLE ROUTINE
380  PLOT X,Y
390  NEXT K
400  VTAB 23
410  GOSUB 1000: REM LOW-RES INFO
420  IF L$ = "S" OR L$ = "s" THEN 5000
425  IF  ASC (L$) = 27 THEN 5000
430  HOME
440  POKE  - 16297,0: REM    SWITCH DISPLAY TO HI-RES
450  HCOLOR= 3
460 X1 = 140:Y1 = 80:P = 1.16:R = 45
470  FOR K = 0 TO 6.4 STEP .05
480  GOSUB 4500: REM     CIRCLE ROUTINE
490  IF K = 0 THEN  HPLOT X,Y: GOTO 510
500  HPLOT  TO X,Y
510  NEXT K
520  VTAB 23: CALL  - 868: REM    CLEAR LINE
530  GOSUB 2000: REM    HI-RES INFO
570 :
580 :
598 :
599 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
798 :
```

```
799 :
800  REM   **--320 MODE CIRCLE--**
801 M = 0: REM   320 Mode
802  GOSUB 7000: REM  Poke SCB's
803  POKE 49193,163: REM  Super Res
804 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM  Move SCB's
806  GOSUB 7500: REM  Poke Color Table
807 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM  Move 320 Color Ta
    ble
809 PI = 3.14159
810 A1 = 0:A2 = 2 * PI:N = 512:R = 80
820 INC = (A2 - A1) / N
825  POKE 8192,64
830  FOR I = A1 TO A2 STEP INC
835 X = R *  SIN (I)
840 Y = R *  COS (I)
845 Y = Y + 50:X = X + 40
850 Y =  INT (Y)
855 Y = Y * 160
860 BP = 15268
865 X = X * .66
870 D = BP + Y + X
875  GOSUB 24000
880 LB = D - 256 * HB
882  POKE 785,LB
883  POKE 789,HB
887  CALL MM
888  NEXT I
889  INPUT A$: GOSUB 6000: REM  Supply text info
890  GOSUB 3000
895  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
898 :
899 :
900  REM   **--640 MODE CIRCLE--**
905 M = 129: REM  640 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM  Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM Move Color Table
940 PI = 3.14159
942 A1 = 0:A2 = 2 * PI:N = 512:R = 80
944 INC = (A2 - A1) / N
945  POKE 8192,4
946  FOR I = A1 TO A2 STEP INC
948 X = R *  SIN (I)
950 Y = R *  COS (I)
```

```
952 Y = Y + 50:X = X + 40
954 Y =  INT (Y)
955 Y = Y * 160
956 BP = 15268
958 X = X * .66
960 D = BP + Y + X
962  GOSUB 24000
964 LB = D - 256 * HB
965  POKE 785,LB
966  POKE 789,HB
970  CALL MM
972  NEXT I
985  INPUT A$: GOSUB 6000: REM Supply text info
990  GOSUB 4000
995  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
997  GOTO 5000
998 :
999 :
1000  REM  **--LOW RESOLUTION--**
1010  INVERSE : PRINT "Low";: NORMAL : PRINT " Resolution--Circle:";: GET
    L$
1020  PRINT
1030  RETURN
1980 :
1990 :
2000  REM  **--HIGH RESOLUTION--**
2010  INVERSE : PRINT "High";: NORMAL : PRINT " Resolution--Circle:";: GET
    L$
2020  PRINT
2030  RETURN
2980 :
2990 :
3000  REM  **-320 Mode Information--**
3010  PRINT "320 Mode Super High Resolution provides for 320 horizontal
    points by "
3020  PRINT
3030  PRINT "200 vertical rows or lines. The 320 points on each line are
     stored in "
3040  PRINT
3050  PRINT "160 bytes. Therefore, one point takes half a byte of memory
    . The"
3060  PRINT
3070  PRINT "characteristics for each line and point are stored in areas
     called Scanline"
3080  PRINT
3090  PRINT "Control Byte memory (SCB's) and Color Table memory."
```

```
3100  VTAB 20: PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
    : INPUT " key to continue.  ";L$
3110  HOME : VTAB 5
3120  PRINT "The next screen displays a circle in 640 Mode!"
3150  RETURN
3998 :
3999 :
4000  REM **--640 Mode Information--**
4010  PRINT "640 Mode Super High Resolution provides for 640 horizontal
    points by"
4020  PRINT
4030  PRINT "200 vertical rows or lines. The 640 points on each line are
    stored in"
4040  PRINT
4050  PRINT "160 bytes just like the 320 Mode. Therefore, one 640 point
    takes one "
4060  PRINT
4070  PRINT "fourth a byte of memory. Again, the characteristics for eac
    h line and"
4080  PRINT
4090  PRINT "point are stored in SCB and Color Table memory."
4100  RETURN
4500  REM     **--CIRCLE ROUTINE--**
4510 X2 = R *  COS (K)
4520 Y2 = R *  COS (K)
4530 X = P * X2 + X1
4540 Y = Y1 - Y2
4550  RETURN
4998 :
4999 :
5000  REM  **--RETURN TO MENU--**
5010  TEXT : HOME : VTAB 5
5020  PRINT "This program has provided a comparison between a low resolu
    tion circle,"
5030  PRINT
5040  PRINT "a high resolution circle, a super high resolution circle in
    320 Mode, and"
5050  PRINT
5060  PRINT "a super high resolution circle in 640 Mode."
5110  POKE  - 16368,0: REM CLEAR KBRD BUFFER
5120  VTAB 20: PRINT "PRESS THE ";: INVERSE : PRINT "RETURN";: NORMAL : INPUT
    " KEY FOR THE MENU. ";L$
5130  END
5990 :
5995 :
6000  REM   **--RETURN TO THE TEXT SCREEN--**
6010  POKE 49193,1: REM  Return to normal display
```

```
6015  TEXT
6020  HOME
6030  VTAB 2
6040  RETURN
6050 :
6060 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA     0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
```

```
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 787,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

BLOAD Instructions for Memory Mover

```
*300.31E
00/0300:A9 00 85 3C A9 20 85 3D A9 7F 85 3E A9 5E 85 3F-)..<) .=)..>)^.?
00/0310:A9 00 85 42 A9 20 85 43 38 20 11 C3 60 00 00-)..B) .C8 .C`..

 300L
1=m   1=x   1=LCbank (0/1)
00/0300: A9 00        LDA #00
00/0302: 85 3C        STA 3C
00/0304: A9 20        LDA #20
00/0306: 85 3D        STA 3D
00/0308: A9 7F        LDA #7F
00/030A: 85 3E        STA 3E
00/030C: A9 5E        LDA #5E
00/030E: 85 3F        STA 3F
00/0310: A9 00        LDA #00
00/0312: 85 42        STA 42
```

```
00/0314: A9 20        LDA #20
00/0316: 85 43        STA 43
00/0318: 38           SEC
00/0319: 20 11 C3     JSR C311
00/031C: 60           RTS
00/031D: 00 00        BRK 00
00/031F: FF FF 00 00  SBC 0000FF,X
00/0323: FF FF 00 00  SBC 0000FF,X
00/0327: FF FF 00 00  SBC 0000FF,X
00/032B: FF FF 00 00  SBC 0000FF,X
```

CHAPTER 11

Advanced IIGS Graphics Information

Before taking a brief look at the TOOLBOX, ROM routines I intend to examine scanline control bytes in greater detail than was possible in the last chapter.

SCANLINE CONTROL BYTES

In Chapter 10, I indicated that a value of 0 produced 320 mode, while a value of 128 indicated 640 mode. Yet we used a value of 129 to actually see examples of 640-mode graphics. As stated in Chapter 10, the additional value was used to indicate that color table 1 (instead of color table 0) was to be used. Actually, a number of different values could have been used to produce the graphics examples provided in the last chapter.

SCB's, or scanline control bytes, provide a number of options to programmers. The eventual value assigned to the SCB is the result of the programmers choices regarding these options. The characteristics of each line (row) of the super high-resolution screen are controlled by a single SCB. That byte (along with other byte characteristics) is stored in an area of memory reserved for the scanline control bytes—E1/9D00. Each line therefore has a single, eight-bit byte of information that indicates to the computer what type of line it should be.

Line Resolution

One of the most important pieces of information contained in each line's characteristics is whether or not the line is to have a resolution of 320 dots or a resolution of 640 dots (pixels). The bits in a byte are numbered from zero to seven, with the zero value on the left and the seven on the right. The bit value that determines the resolution of the line is number seven. If the 7th bit contains a zero, the resolution of that line will be 320 pixels. If the 7th bit

contains a one, the resolution of that line will be 640 pixels. As in just about all cases involving hexadecimal numbers, the last four bits are considered the high-order bits and placed first when writing the hexadecimal number. Therefore, we have the following:

A bit value of 0 in bit 7 indicates 320 mode
A bit value of 1 in bit 7 indicates 640 mode

Assuming for the moment that all other bit values are zero, you would have the following bit pattern for the two modes:

Mode	*Binary*	*Inverted*	*Hex*	*Decimal*
320 mode	0000 0000	0000 0000	0 0	0
640 mode	0000 0001	1000 0000	8 0	128

Refer back to the binary-to-hex conversion chart provided in Chapter 8 if you need to. As you can see, the decimal values of zero and 128 were chosen as only the first of many values that can represent the desired resolution. For example, the use of decimal 129 produces the following:

Mode	*Binary*	*Inverted*	*Hex*	*Decimal*
640	1000 0001	1000 0001	81	129

A decimal value of 127 would indicate that the line should contain 320 pixels:

Mode	*Binary*	*Inverted*	*Hex*	*Decimal*
320	1111 1110	0111 1111	7F	127

The single zero in the binary definition of this byte occurs in the 7th bit position (reading left to right and starting with bit zero). That position controls the mode of resolution used on that line. Therefore, any decimal value below 128 will indicate 320-mode resolution, and any decimal value 128 or greater will signify 640-mode resolution.

Color-Table Designation

Each of the other bit positions provide additional definition to the characteristics of the line. The first four bit positions indicate which color table the line is to use to obtain the colors for its individual points. Since each color table can have 16 colors, each line can have 16 colors for any of its points. (As noted in Chapter 10, 640 mode handles color in a different way. I will discuss 640 mode further a little later.) I also said that super high-resolution can have 16 color tables in memory at one time. Therefore, each SCB must indicate which color table the line will use. If the value of bit position zero is zero, and the other three bit positions are also zeros, then that line will get the colors for its individual points from color table zero. If the value of bit position zero is a one and the other three bit positions are zeros,

then that line will get the colors for its individual points from color table 1. The following examples should help explain. Only bits zero to three are defined, the other four bits of a scanline control byte are not used for color-table definition.

Binary	*Inverted*	*Hex*	*Decimal*	*Table*
0000	0000	0	0	0
1000	0001	1	1	1
1010	0101	5	5	5
0011	1100	C	12	12
1111	1111	F	15	15

SCB Bit 4

We now have five of the eight bit positions in a scanline control byte defined. Bit position number four is very easy to understand also. It must always contain a value of zero. This bit position is "reserved" by the manufacturer, perhaps for future use.

SCB Bit 6 – Interrupts

The last two bit positions are somewhat more complicated. Bit number six is called an "interrupt." If the value of this bit is a one, then each time the screen is "refreshed" (the electrical current completes its cycle) an interrupt is generated for that line. Through assembly-language programming, certain special effects can be produced with the use of interrupts. All the example programs included in this book set bit six to zero.

SCB Bit 5 – Fill Mode

The last bit position to be explained, position number five, is perhaps the most interesting and challenging, so please read carefully. If the value in bit five is a one, a special super high-resolution mode goes into effect (is enabled). This mode is called the "**fill**" mode. First of all, this option is only available in 320-point resolution. Enabling the fill mode in 640-point resolution can cause unpredictable results. Therefore, whenever you have a bit value of one in the 7th bit position (indicating 640-mode resolution), you should not have a value of one in the 5th bit position (indicating enablement of the fill option). When translated to decimal values, this means that certain values above 128 *should not* be used for a scanline control byte. None of the values below 128 decimal are a problem, since any value below 128 indicates 320-point resolution.

Fill mode in 320-point resolution can produce stunning graphic images, but I have found it difficult to accurately predict and produce the desired results. The fill mode operates by filling any pixel that has a zero value with the color of the pixel (point) to the left of it. The zero-value pixel retains its zero, so that the color of the filled area will change

if the nonzero pixel changes color. The obvious problem in this mode is how to limit the zero areas. If you start out with a clear screen—a screen full of zeros—it would seem impossible to position a nonzero pixel anywhere without automatically filling every zero-value pixel to the right of it.

One answer appears to be to use a separate color table for the fill mode. But in fact, if you use an SCB setting with either table presented in Chapter 10, you will experience another rather strange phenomenon associated with the fill mode. As soon as the color table has been defined, the entire screen changes color (with the color depending upon the color table used). Yet the value in every screen memory location remains zero. This problem disappears though, if you alter the color table slightly.

Certain color values apparently act as blocking agents; i.e., they do not fill zero-value pixels to their right, yet do act as the end point for fill areas to their left. In the table presented in this chapter, and in the two tables listed in the last chapter, the color value two and every 16th value thereafter seem to act as blocking agents. I have not been able to check all these numbers in every situation, but generally, using these two tables, the numbers I indicated can be used as right-hand end points. The problem of an unwanted colored background can be avoided by using this color table:

```
9700 REM **--FILL COLOR TABLE--**
9710 DATA 0,0,119,7,0,0,44,7,15,0,128,0,112,15,0,13
9720 DATA 169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
```

For straight-line drawing, the blocking-agent values work fine by themselves, but curves may require more than one layer of blocking agents. The FILL.CIRCLE program listed at the end of this chapter actually draws two outside circles, using a color value of two for both circles. Inside both of these circles, a third circle is drawn using a color value that will fill the zero-value pixels. This third circle is also drawn on a one-cycle delay to ensure that the outside circles have completely blocked the inside circle from filling areas that it should not fill. The routine is slow, but at least it does work, and it demonstrates that even the fill mode can be used from BASIC. Figures 11.1 through 11.4 show various shapes produced in the Fill mode.

There may be a better way of obtaining the fill effect, but as of this publication, I have not found one. The term "blocking agent" (used to describe certain colors that do not fill zero pixels to their right) is unique to this book. In other publications, you may come across other terms that essentially describe the same effect. Relatively little has been written so far about the fill option, and what has been published deals with the fill option in the context of assembly-language programming rather than Applesoft BASIC programming.

SCB Summary

The following table summarizes the allocations for the bit patterns in the scanline control bytes:

Bit #	*0 Value*	*1 Value*
7	320 res.	640 res.
6	no interrupt	interrupt enabled
5	no fill	fill mode enabled
4	must be set to 0	
0–3	color table indication	

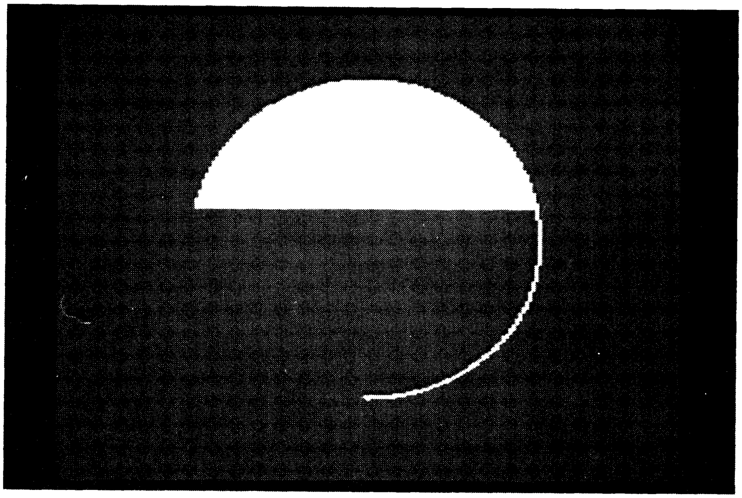


FIGURE 11.1. Fill effect in progress.

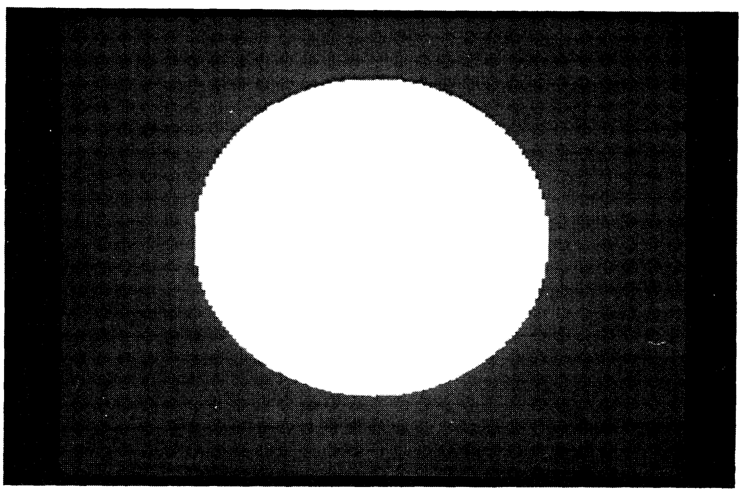


FIGURE 11.2. Super-resolution circle in fill mode.

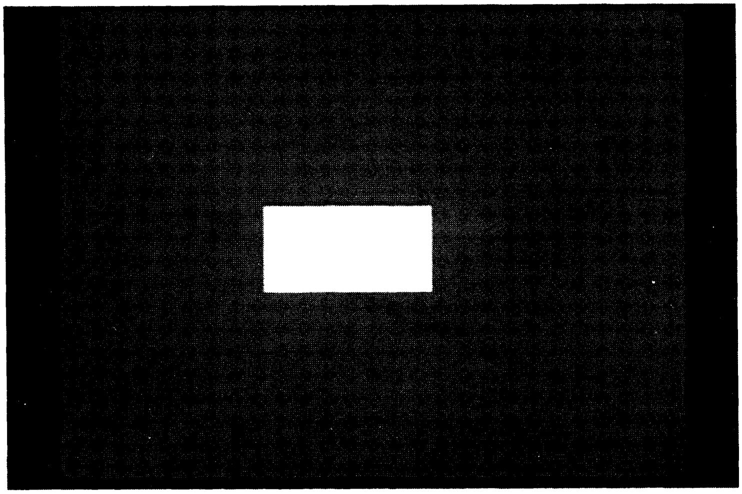


FIGURE 11.3. Super-resolution rectangle in fill mode.

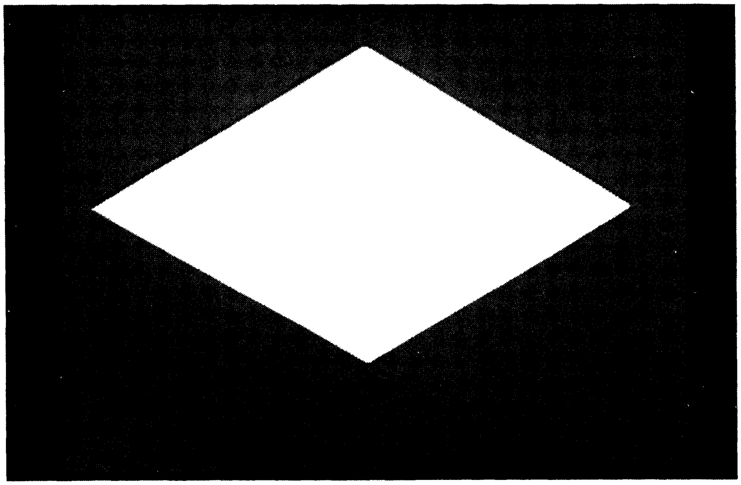


FIGURE 11.4. Super-resolution diamond in two-color fill mode.

A few examples should help explain how to use this information.

The simplest example is to use 320 mode, without the fill option, without interrupts and to use the first color table, color table 0. 320 mode equals a zero in the 7th bit position. Bits five and six equal zero also, since we do not want the fill mode or interrupts. Bit four is always a zero and the four-bit value of color table 0 is zero. Therefore, all eight bits are zeros, the hex value is a zero, and the decimal value is a zero. This zero value must be placed in

the scanline control byte location of each line that is to be defined in this manner. If all the lines on the screen are to be characterized by this definition, a loop can be used to place a value of zero in the scanline control byte memory area for all 200 lines.

If you want to define a line as having a 640-point resolution, bit seven must contain the value of one. Bits four, five, and six will be zeros since bit four is always a zero and bits five and six either cannot be implemented in 640 mode or cannot be used effectively with BASIC. The only remaining decision is which table to draw color values from. If you have created a 640-mode color table and have stored it as the third color table in the color-table memory area (color table 2), you get the following bit pattern for the scanline control byte definition—reading left to right, zero to seven:

Bit #	0123	4567
Binary	0100	0001

Inverted for easier conversion to a hexadecimal value, the bit pattern looks like this:

Bit #	7654	3210
Binary	1000	0010

Translating each of the four bits into its hexadecimal value results in

Bit #	7654	3210	
Binary	1000	0010	
Hex	8	2	or 82

Finally, converting hex 82 to a decimal value yields 130. The decimal value 130 should be entered as the scanline control byte definition for a line in 640 mode, without interrupts or fill mode, that uses color table 2 for its pixel color values.

One more example should be sufficient, this time using the fill mode. Therefore, you know that you must work with 320 mode (zero in bit seven and one in bit five). You do not want the interrupt option and will be using color table one in bit zero, the second color table in the color table memory location. The binary definition is:

Bit #	0123	4567
Binary	1000	0100

Inverted, the pattern equals

Bit #	7654	3210
Binary	0010	0001

The same pattern in hexadecimal is

Binary	0010	0001
Hex	2	1

and translated to decimal:

Hex	21
Decimal	33

Therefore, the decimal value 33 should be entered for a SCB definition of a line (or lines) that is in the 320-resolution fill mode without interrupts and using color table 1. Once you have worked with SCBs for a brief time, you should find that they are not difficult to understand or hard to change.

640-MODE COLOR-TABLE DEFINITIONS

As noted in Chapter 10, 640 mode uses colors in a different way than 320 mode does. The reason for this is that 320 mode has four bits available to it for pixel color identification, while 640 mode only has two bits available. Each line, regardless of the resolution, has 160 bytes, and each byte contains eight bits. Therefore, in 320-resolution mode, each pixel can use up to four of those bits for its color identification. A four-bit identification means that each pixel can be one of 16 different colors. However, in 640 mode, each pixel can use only two bits for its color identification. A two-bit identification means that each pixel can be one of only four different colors. Yet, the color tables are set up for 16 colors. The question is, which four colors in a 16-color color table should the two bits refer to? Instead of making it easy for us, Apple decided to add flexibility, so please read the next section carefully.

The position of the pixel's two bits within the full byte's bit pattern determines which four colors are possible for the single 640-resolution pixel.

Bit #	0 1	2 3	4 5	6 7

If the two bits appear as the first two bits within the byte (bits number zero and one) then the four colors available are the last four colors in the color table (the first shall be last?). If the two bits appear as bits number two and three then the four colors available are the colors numbered 8, 9, 10 and 11 in the color table. If the two bits appear as bits number four and five, then the four colors available are the colors numbered four, five, six and seven in the color table. If the two bits appear as bits numbered six and seven then the four colors available are the colors numbered zero, one, two and three in the color table. Obviously, then every pixel cannot display all of the 16 colors in the color table.

In practice, this limitation is not too severe. First, for those using BASIC to produce graphics on the super high-resolution screen, the smallest amount that can be plotted is one byte of information. Therefore, in 320-resolution mode, each byte plots two points and in

640-resolution mode, each byte plots four points or pixels. Regardless of the mode, each byte has 256 different combinations of colors that it can plot. If you know what you are doing, you can simulate more than four-color capability in 640 mode through a process called "dithering." Dithering alternates the values of the colors in pixels that are next to each other, giving the appearance of more than four-colors-per-pixel capability.

As you can see, you can spend a great deal of time just learning about the many things that the IIGS is capable of, even if you are using an unenhanced BASIC. However, the real power of IIGS graphics is stored inside the computer in ROM.

THE TOOLBOX OR TOOLSET

With the introduction of the Macintosh, Apple settled on a specific way of doing things on Apple computers by creating a very complex series of routines that allow standardization of the way tasks are accomplished on their computers. Sometimes referred to as the Macintosh User Interface, these routines have now been transported to the Apple *II* line through the Apple IIGS.

Before the introduction of the Macintosh User Interface, there were many many different methods used by programmers to allow users to make choices from a menu of available options. Some programmers used numbers for the options, others used letters, and others used both numbers and letters. Still others used neither numbers or letters, preferring to simply highlight different phrases that indicated the choices available. Even the method of moving the highlight varied from program to program. The positioning of the choices on the screen depended upon the programmer's preference and sometimes was not even consistent within a single system of menus. Individual companies tried to standardize their own programs but were often unsuccessful. Industry-wide standardization was out of the question.

With the introduction of the Macintosh User Interface, Apple virtually eliminated diversity in the methods by which programs communicated with their users on the Macintosh computer. Now, all programs on the Macintosh operate in much the same way. The choice of options, for example, is always at the top of the screen on a menu bar. Choices are made from the menu by pointing and clicking the mouse (although some selections can still be made with the keyboard).

To force all programmers into doing things Apple's way, Apple created many routines that greatly aided in the creation of application programs *if* the program was designed to be consistent with the Macintosh User Interface guidelines. If the programmer chooses to do things differently, many of the routines cannot be used because certain "interdependencies" exist between the routines. In other words, if you do not want to use routine A but do want to use routine B, you cannot use B if it is interdependent with A. In effect, Apple told programmers to do it Apple's way or else create code from the ground up—including the code to display a single character, since all text on the Macintosh is graphics based.

Apple has now moved the Macintosh User Interface philosophy and approach over to the Apple *II* line of computers on the Apple IIGS. As long as programmers stay within the limits of the old eight-bit capabilities of the Apple *II* line, they can do things however they choose. But if programmers, even part-time or leisure-time programmers, want to create something in the 16-bit world of the Apple IIGS, they must do things Apple's way or develop methods and routines from scratch – including text display. This restriction even includes the languages that can be used to access the Apple routines. As indicated in Chapter 4, Apple decided not to enhance Applesoft BASIC to use the routines created to assist programmer development. (Several new versions of BASIC from different companies may soon be available. These versions may or may not be able to access all of the ROM routines.)

In the first part of this chapter and in Chapter 10, I have shown how BASIC can be used to create graphic displays on the super high-resolution screen. However, the real power of this screen lies in the 128K or so of code that has been developed and tested by Apple and that resides in the machine's ROM – code that, for all practical purposes, is not available to BASIC programmers.

Through the development of special assembly-language routines (like the one presented in Chapter 10), many of these ROM routines should eventually become available to BASIC programmers, but even then the interdependencies between the routines will force the use of a very specific type of programming that requires a great deal of time to learn and understand. Kyle Maskim, Director of Product Development for CLARIS (Apple's independent software company), stated it this way: "It takes time to develop the taste for the interface." (Outside Apple, November 1987, p.3). By inference, this means that few non-professional programmers are going to spend the time to learn all that is necessary to fully use the power of the ROM routines. However, once you have learned this type of programming (sometimes called circular programming), new applications are created by much the same process as before.

The heart of this new type of programming is in the routines collectively called the toolset or TOOLBOX. When the IIGS was introduced, Apple also introduced approximately 128K of routines that touch upon just about every task imaginable. The collection of routines is divided into groups of related routines: memory-management routines, graphics-management routines, file-management routines, menu-management routines, event-management routines, etc. Not surprisingly, these collections are referred to as managers: memory manager, file manager, menu manager, event manager, etc. The graphics-management collection is called Quickdraw II. (This collection is called Quickdraw on the Macintosh.)

The Quickdraw II routines allow an assembly-language programmer to accomplish everything that has been covered in Chapter 10 and in this chapter with relatively few instructions. However, the interdependencies between Quickdraw II and the other managers significantly add to the complexity of creating the super high-resolution graphics you have seen so far. To go beyond what I have demonstrated to this point is problematic. The job of placing text on the super high-resolution screen provides a good example.

I have not yet been able to figure a way of labeling graphic material on the super high-resolution screen without either (1) creating a rather long assembly-language routine that uses Quickdraw II (and thus certain other managers also) or (2) developing a complete relocatable ASCII character set using BASIC instructions. Neither solution is possible within the scope of this book. Therefore, I am left with the unpalatable task of recommending that you learn 65816 Assembly Language and how to use the different TOOLBOX managers if you want to accomplish much more than has been presented in these chapters on the IIGS. Two other possibilities are open to you: wait for others to develop additional routines that access the Quickdraw II routines, or develop such routines yourself. Listed below (in startup order) is the latest information regarding the level of interdependencies that exist between the various managers in the toolset:

Tool Locator (#1)
Memory Manager (#2)
Miscellaneous Tools (#3)
QuickDraw II (#4)
Event Manager (#6)
Window Manager (#14)
Control Manager (#16)
Menu Manager (#15)
LineEdit (#20)
Dialog Manager (#21)
Sound Tools (#8)
Note Synthesizer (#25)
Standard File Operations (#23)
Scrap Manager (#22)
Desk Manager (#5)
List Manager (#28)
Print Manager (#19)
Font Manager (#27)

Apple provided this recommended startup order because of the circular dependencies between the various toolsets. As you can see, just learning about these toolsets (let alone learning to use them) could take a long, long time.

Writing this book of graphics has been an exhilarating, frustrating, and humbling experience. It has been especially humbling because I am acutely aware that there are many paid and unpaid programmers who know a great deal about Apple *II* graphics. It is frustrating because, in a very real sense, there is no good point at which to stop—I could write much more concerning just the standard Apple *II* screens. Even more frustrating is the knowledge that, given time, additional super high-resolution routines could be developed—routines that would be of value to you. Nevertheless, I have come to the conclusion of the programs developed for this book. It is up to you to develop additional super

high-resolution routines and programs. The challenge now is to see how far we can go using Applesoft BASIC to create super high-resolution graphics.

REVIEW QUESTIONS

1. Which bit position in a scanline control byte controls the resolution mode?
2. Which bit position in a scanline control byte should always contain a zero value?
3. True or False? Decimal values above 128 in an SCB definition indicate 320 mode.
4. True or False? The fill option is available for both 320-resolution and 640-resolution modes.
5. True or False? Both 320 mode and 640 mode specify color for individual pixels in the same way.
6. What is the combined set of 128K ROM routines called?

Fill Rectangle

```
100  REM  ***--FILL.RECTANGLE--***
110 :
120 :
500  REM   **--Title--**
510  HOME : VTAB 5: HTAB 35
520  INVERSE : PRINT "Fill Rectangle": NORMAL
530  VTAB 22: HTAB 28
540  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
550  INPUT " key: ";NUL$
560 :
570 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
730 :
740 MESSAGE$ = "320 Mode Filled Rectangle."
750  GOSUB 3000: REM  Message routine
798 :
799 :
800  REM    **--320 MODE--**
805 M = 32: REM 320 Mode/With Fill
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color
    Table
835  GOSUB 15000: REM Clear Screen
840  POKE 8192,66: REM  set color of point
842  FOR I = 80 TO 120
843  POKE 8192,2: REM    set color of point
845 ROW = I:COL = 100: GOSUB 22000:D = ML: GOSUB 24000
850  POKE 785,LB: POKE 789,HB: REM  Set new destination address
855  CALL MM: REM Move Memory
857  POKE 8192,7: REM    set color of point
860 ROW = I:COL = 50: GOSUB 22000:D = ML: GOSUB 24000
865  POKE 785,LB: POKE 789,HB: REM   Set new destination address
870  CALL MM: REM  Move Memory
880  NEXT I
890  INPUT " ";NUL$
898 :
899 :
900  REM   **--640 MODE--**
905 :
910 :
```

```
915  REM   *********************
920  REM
925  REM   NO FILL IN 640 MODE
930  REM
935  REM   *********************
980 :
990 :
2000  REM    **--End--**
2010  POKE 49193,1: REM   Switch to text
2020  HOME : VTAB 10
2030  PRINT "This has been a demonstration of Fill Mode."
2040  END
2050 :
2060 :
3000  REM   **--Message On Text Page--**
3010  POKE 49193,1: REM   Return to text page
3020  HOME : VTAB 10
3030  PRINT MESSAGE$
3040  VTAB 22
3050  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
3060  INPUT " key: ";NUL$
3070  POKE 49193,163
3080  RETURN
3090 :
3100 :
7000  REM **--SCANLINE CONTROL BYTES--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
```

```
9700  REM    **--320 Mode Fill Color Table--**
9710  DATA     0,0,119,7,0,0,44,7,15,0,128,0,112,15,0,13
9720  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9730 :
9740 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
22000  REM  **--Calculate Memory Location From Col.& Row Value--**
22010 ML = 8192 + ((ROW - 1) * 160) + COL
22020  RETURN
22030 :
22040 :
```

```
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Fill Diamond

```
100  REM    ***--FILL.DIAMOND--***
110 :
120 :
500  REM   **--Title--**
510  HOME : VTAB 5: HTAB 35
520  INVERSE : PRINT "Filled Diamond": NORMAL
530  VTAB 22: HTAB 28
540  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
550  INPUT " key: ";NUL$
560 :
570 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
798 :
799 :
800  REM    **--320 MODE--**
805 M = 32: REM 320 Mode/With Fill
810  GOSUB 7000: REM Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color
    Table
835  GOSUB 15000: REM Clear Screen
840  FOR I = 0 TO 159
842  IF I > 79 THEN 860
843  POKE 8192,2: REM  Set color of point
845 ROW = I:COL = 80 + I: GOSUB 22000:D = ML: GOSUB 24000
847  POKE 785,LB: POKE 789,HB: REM   Set new destination address
849  CALL MM: REM  Move Memory
850  POKE 785,LB: POKE 789,HB: REM  Set new destination address
851  POKE 8192,6: REM  Set color of point
853 ROW = I:COL = 80 - I: GOSUB 22000:D = ML: GOSUB 24000
855  POKE 785,LB: POKE 789,HB: REM    Set new destination address
857  CALL MM: REM   Move Memory
859  GOTO 880
860  POKE 8192,2: REM  Set color of point
862 ROW = I:COL = 160 - (I - 80): GOSUB 22000:D = ML: GOSUB 24000
```

```
864  POKE 785,LB: POKE 789,HB: REM   Set new destination address
866  CALL MM: REM  Move Memory
868  POKE 8192,5: REM  Set color of point
870 ROW = I:COL = I - 80: GOSUB 22000:D = ML: GOSUB 24000
872  POKE 785,LB: POKE 789,HB: REM    Set new destination address
874  CALL MM: REM   Move Memory
880  NEXT I
890  INPUT " ";NUL$
898 :
899 :
900  REM   **--640 MODE--**
905 :
910 :
915  REM  **********************
920  REM
925  REM  NO FILL IN 640 MODE
930  REM
935  REM  **********************
980 :
990 :
2000  REM    **--End--**
2010  POKE 49193,1: REM    Switch to text
2020  HOME : VTAB 10
2030  PRINT "This has been a demonstration of the Fill Mode."
2040  END
2050 :
2060 :
3000  REM   **--Message On Text Page--**
3010  POKE 49193,1: REM   Return to text page
3020  HOME : VTAB 10
3030  PRINT MESSAGE$
3040  VTAB 22
3050  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
3060  INPUT " key: ";NUL$
3070  POKE 49193,163
3080  RETURN
3090 :
3100 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
```

```
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9700  REM    **--320 Mode Fill Color Table--**
9710  DATA     0,0,119,7,0,0,44,7,15,0,128,0,112,15,0,13
9720  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9730 :
9740 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
```

```
17050  CALL MM
17060  RETURN
17070 :
17080 :
22000  REM  **--Calculate Memory Location From Col.& Row Value--**
22010 ML = 8192 + ((ROW - 1) * 160) + COL
22020  RETURN
22030 :
22040 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Fill Circle

```
100  REM  ***--FILL.CIRCLE--***
110 :
120 :
500  REM   **--Title--**
510  HOME : VTAB 5: HTAB 35
520  INVERSE : PRINT "Filled Circle": NORMAL
530  VTAB 22: HTAB 28
540  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
550  INPUT " key: ";NUL$
560 :
570 :
700  REM  **--Super Res Initialization--**
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & E1
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
798 :
799 :
800  REM  **--320 MODE--**
801 M = 32: REM    320 Mode
802  GOSUB 7000: REM  Poke SCB's
803  POKE 49193,163: REM  Super Res
804 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM  Move SCB's
806  GOSUB 7500: REM  Poke Color Table
807 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM  Move 320 Color Ta
    ble
808  GOSUB 15000
809 PI = 3.14159
810 A1 = 0:A2 = 2 * PI:N = 512:R = 80:R2 = 81:R3 = 79
820 INC = (A2 - A1) / N
830  FOR I = A1 TO A2 STEP INC
831 :
```

```
832  REM   *--INSIDE BARRIER CIRCLE--**
833  POKE 8192,2
835 X = R *  SIN (I)
840 Y = R *  COS (I)
845 Y = Y + 50:X = X + 40
850 Y =  INT (Y)
855 Y = Y * 160
860 BP = 15288: REM BEGINNING POINT
865 X = X * .66
870 D = BP + Y + X
875  GOSUB 24000
880 LB = D - 256 * HB
882  POKE 785,LB
883  POKE 789,HB
887  CALL MM
890 :
900  REM  *-OUTSIDE BARRIER CIRCLE--*
905 X2 = R2 *  SIN (I)
910 Y2 = R2 *  COS (I)
915 Y2 = Y2 + 50:X2 = X2 + 40
920 Y2 =  INT (Y2)
925 Y2 = Y2 * 160
930 BP = 15288
935 X2 = X2 * .66
940 D = BP + Y2 + X2
945  GOSUB 24000
946 LB = D - 256 * HB
947  POKE 785,LB
948  POKE 789,HB
949  CALL MM
950 :
951  REM  *--ACTUAL FILLED CIRCLE--**
952  POKE 8192,7
953 Q = I - INC
955 X3 = R3 *  SIN (Q)
960 Y3 = R3 *  COS (Q)
965 Y3 = Y3 + 50:X3 = X3 + 40
970 Y3 =  INT (Y3)
975 Y3 = Y3 * 160
980 BP = 15288
985 X3 = X3 * .66
990 D = BP + Y3 + X3
995  GOSUB 24000
996 LB = D - 256 * HB
997  POKE 785,LB
998  POKE 789,HB
999  CALL MM
1000 :
```

```
1100  NEXT I
1200 :
1300 :
1500  INPUT " ";NUL$
1600 :
1700 :
2000  REM    **--End--**
2010  POKE 49193,1: REM    Switch to text
2020  HOME : VTAB 10
2030  PRINT "This has been a demonstration of the Fill Mode"
2040  END
2050 :
2060 :
7000  REM  **--Scanline Control Bytes--**
7010  FOR I = 0 TO 199
7020  POKE 8192 + I,M
7030  NEXT I
7040  RETURN
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9700  REM    **--320 Mode Fill Color Table--**
9710  DATA     0,0,119,7,0,0,44,7,15,0,128,0,112,15,0,13
9720  DATA     169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9730 :
9740 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
```

```
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Mode Switching

```
100  REM    ***--MODE.SWITCHING--***
110 :
120 :
500  REM  **--Title--**
510  HOME : VTAB 5: HTAB 35
520  INVERSE : PRINT "Mode Switching": NORMAL
530  VTAB 22: HTAB 28
540  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
550  INPUT " key: ";NUL$
560 :
570 :
700  REM  **--Super Res Initialization--**
```

```
710  POKE 49205,0: REM  Initialize Shadowing Memory to Banks E0 & El
720 MM = 768: REM  Move Memory--Location of Assembly Lang.routine
725  GOSUB 15000: REM  Clear Screen
730 MESSAGE$ = "320 Mode diamond with different colors."
740  GOSUB 3000: REM  message subroutine
798 :
799 :
800  REM    **--320 MODE--**
805 M = 0: REM   320 Mode
810  GOSUB 7000: REM  Poke SCB's
815  POKE 49193,163: REM Super Res
820 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
825  GOSUB 7500: REM Poke Color Table
830 EH = 32:EL = 31:DH = 158:DL = 0: GOSUB 17000: REM Move 320 Color Tab
    le
835  GOSUB 15000: REM Clear Screen
840  FOR I = 1 TO 159
842  IF I > 79 THEN 860
843  POKE 8192,102: REM Set color of point
845 ROW = I:COL = 79 + I: GOSUB 22000:D = ML: GOSUB 24000
847  POKE 785,LB: POKE 789,HB: REM Set new destination address
849  CALL MM: REM  Move Memory
851  POKE 8192,153: REM  Set color of point
853 ROW = I:COL = 79 - I: GOSUB 22000:D = ML: GOSUB 24000
855  POKE 785,LB: POKE 789,HB: REM Set new destination address
857  CALL MM: REM   Move Memory
859  GOTO 880
860  POKE 8192,112: REM   Set color of point
862 ROW = I:COL = 160 - (I - 79): GOSUB 22000:D = ML: GOSUB 24000
864  POKE 785,LB: POKE 789,HB: REM Set new destination address
866  CALL MM: REM  Move Memory
868  POKE 8192,5: REM   Set color of point
870 ROW = I:COL = I - 79: GOSUB 22000:D = ML: GOSUB 24000
872  POKE 785,LB: POKE 789,HB: REM Set new destination address
874  CALL MM: REM Move Memory
880  NEXT I
890  INPUT " ";NUL$
898 :
899 :
900  REM   **--640 MODE--**
902 MESSAGE$ = "640 Mode diamond with different colors."
903  GOSUB 3000: REM   message subroutine
905 M = 129: REM 604 Mode
910  GOSUB 7000: REM Poke SCB's
915  POKE 49193,163: REM  Super Res
920 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
925  GOSUB 7500: REM Poke Color Table
930 EH = 32:EL = 31:DH = 158:DL = 32: GOSUB 17000: REM Move Color Table
```

```
935  GOSUB 15000: REM Clear Screen
940  FOR I = 1 TO 159
942  IF I > 79 THEN 960
943  POKE 8192,4: REM  Set color of point
945 ROW = I:COL = 79 + I: GOSUB 22000:D = ML: GOSUB 24000
947  POKE 785,LB: POKE 789,HB: REM  Set new destination address
949  CALL MM: REM  Move Memory
951  POKE 8192,5: REM Set color of point
953 ROW = I:COL = 79 - I: GOSUB 22000:D = ML: GOSUB 24000
955  POKE 785,LB: POKE 789,HB: REM  Set new destination address
957  CALL MM: REM  Move Memory
959  GOTO 980
960  POKE 8192,6: REM Set color of point
962 ROW = I:COL = 160 - (I - 79): GOSUB 22000:D = ML: GOSUB 24000
964  POKE 785,LB: POKE 789,HB: REM  Set new destination address
966  CALL MM: REM Move Memory
968  POKE 8192,7: REM Set color of point
970 ROW = I:COL = I - 79: GOSUB 22000:D = ML: GOSUB 24000
972  POKE 785,LB: POKE 789,HB: REM  Set new destination address
974  CALL MM: REM  Move Memory
980  NEXT I
990  INPUT " ";NUL$
998 :
999 :
1000  REM **--320 Mode With Fill Option--**
1002 MESSAGE$ = "320 Mode diamond in fill mode w/color table #0"
1003  GOSUB 3000: REM    message subroutine
1010 M = 32: REM  320 Mode/Color Table #0
1020  GOSUB 7000: REM Poke SCB's
1030 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM  Move SCB's
1040  INPUT " ";NUL$
1050 :
1060 :
1100  REM  **--320 Mode--**
1102 MESSAGE$ = "320 Mode diamond with different colors."
1103  GOSUB 3000: REM   message subroutine
1110 M = 0: REM  320 Mode
1120  GOSUB 7000: REM Poke SCB's
1130 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
1140  INPUT " ";NUL$
1150 :
1160 :
1200  REM  **--640 Mode--**
1202 MESSAGE$ = "640 Mode diamond with different colors."
1203  GOSUB 3000: REM    message subroutine
1210 M = 129: REM   640 Mode
1220  GOSUB 7000: REM  Poke SCB's
1230 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM Move SCB's
```

```
1240  INPUT " ";NUL$
1250 :
1260 :
1300  REM   **--320 Mode With Fill Option--**
1302 MESSAGE$ = "320 Mode diamond in fill mode w/color table #1"
1303  GOSUB 3000: REM message subroutine
1310 M = 33: REM   320 Mode/Color Table #1
1320  GOSUB 7000: REM   Poke SCB's
1330 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM  Move SCB's
1340  INPUT " ";NUL$
1350 :
1360 :
1400  REM    **--320 Mode With Fill Option--**
1402 MESSAGE$ = "320 Mode diamond in fill mode w/color table #2"
1403  GOSUB 3000: REM  message subroutine
1410 M = 34: REM    320 Mode/Color Table #2
1420  GOSUB 7000: REM    Poke SCB's
1430 EH = 32:EL = 199:DH = 157:DL = 0: GOSUB 17000: REM   Move SCB's
1440  GOSUB 7500: REM  Poke Color Table
1450 EH = 32:EL = 31:DH = 158:DL = 64: GOSUB 17000: REM Move Color
    Table
1460  INPUT " ";NUL$
1470 :
1480 :
2000  REM   **--End--**
2010  POKE 49193,1: REM  return to text screen
2020  HOME : VTAB 10
2030  PRINT "This has been a demonstration of the effects when only the
    SCB's are changed."
2040  END
2050 :
2060 :
3000  REM  **--Message On Text Page--**
3010  POKE 49193,1: REM  Return to text page
3020  HOME : VTAB 10
3030  PRINT MESSAGE$
3040  VTAB 22
3050  PRINT "Please press the ";: INVERSE : PRINT "RETURN";: NORMAL
3060  INPUT " key: ";NUL$
3070  POKE 49193,163
3080  RETURN
3090 :
3100 :
7000  REM     **--SCANLINE CONTROL BYTES--**
7010  FOR I = 0 TO 199
7020  POKE 8192 * I,M
7030  NEXT I
7040  RETURN
```

```
7050 :
7060 :
7500  REM   **--COLOR TABLE--**
7510  FOR I = 0 TO 31
7520  READ CT
7530  POKE 8192 + I,CT
7540  NEXT I
7550  RETURN
7560 :
7570 :
8000  REM  **--CLEAR HIRES MEMORY--**
8010  POKE 230,32: CALL  - 3086
8020  POKE 230,64: CALL  - 3086
8040  RETURN
8050 :
8060 :
9000  REM     **--320 Mode Standard Color Table--**
9010  DATA      0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA      169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9030 :
9040 :
9500  REM     **--640 Mode Standard Color Table--**
9510  DATA      0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA      0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
9700  REM     **--320 Mode Fill Color Table--**
9710  DATA      0,0,119,7,0,0,44,7,15,0,128,0,112,15,0,13
9720  DATA      169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9730 :
9740 :
15000  REM  **--CLEAR MEMORY AND SUPER-RES SCREEN--**
15002 :
15005  REM  *-Poke Ending Address-*
15010 EA = 24192: REM  Ending Address
15020 D = EA
15030  GOSUB 24000: REM  Calculate Low and High Byte
15040  POKE 777,LB: REM  Poke Low Byte
15050  POKE 781,HB: REM  Poke High Byte
15052 :
15055  REM  *-Poke Destination Address-*
15060 DA = 8192: REM  Destination Address
15070 D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15080  GOSUB 8000: REM  Clear Hires Screens
15120  CALL MM: REM   Move Info to E1 bank
15130 :
15200  REM  *-Poke Second Ending Address-*
15210 EA = 24192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
```

```
15220 :
15250  REM  *-Poke Second Destination Address-*
15260 DA = 24191:D = DA: GOSUB 24000: POKE 785,LB: POKE 789,HB
15310  GOSUB 8000: CALL MM
15330 :
15500  REM  *-Reset Routine For One Address-*
15510 EA = 8192:D = EA: GOSUB 24000: POKE 777,LB: POKE 781,HB
15560  RETURN
15570 :
15580 :
17000  REM   **--MEMORY MOVER ADDRESSES--**
17010  POKE 777,EL: REM   Ending Low Byte
17020  POKE 781,EH: REM   Ending High Byte
17030  POKE 785,DL: REM   Destination Low Byte
17040  POKE 789,DH: REM   Destination High Byte
17050  CALL MM
17060  RETURN
17070 :
17080 :
22000  REM  **--Calculate Memory Location From Col.& Row Value--**
22010 ML = 8192 + ((ROW - 1) * 160) + COL
22020  RETURN
22030 :
22040 :
24000  REM   **--CALCULATE HIGH & LOW BYTE--**
24010 HB =  INT (D / 256)
24020 LB = D - 256 * HB
24030  RETURN
```

Color Tables

```
9000  REM   **--320 Mode Standard Color Table--**
9010  DATA    0,0,119,7,65,8,44,7,15,0,128,0,112,15,0,13
9020  DATA    169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9030 :
9040 :
9500  REM   **--640 Mode Standard Color Table--**
9510  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9520  DATA    0,0,0,15,240,0,255,15,0,0,15,0,240,15,255,15
9540 :
9550 :
9700  REM   **--320 Mode Fill Color Table--**
9710  DATA    0,0,119,7,0,0,44,7,15,0,128,0,112,15,0,13
9720  DATA    169,15,240,15,224,0,223,4,175,13,143,7,204,12,255,15
9730 :
9740 :
```

BLOAD Instructions for Memory Mover

```
CALL-151

*300.31E
00/0300:A9 00 85 3C A9 20 85 3D A9 7F 85 3E A9 5E 85 3F-)..<) .=)..>>^.?
00/0310:A9 00 85 42 A9 20 85 43 38 20 11 C3 60 00 00-)..B) .C8 .C`..
*

 300L
1=m   1=x   1=LCbank (0/1)

00/0300: A9 00       LDA #00
00/0302: 85 3C       STA 3C
00/0304: A9 20       LDA #20
00/0306: 85 3D       STA 3D
00/0308: A9 7F       LDA #7F
00/030A: 85 3E       STA 3E
00/030C: A9 5E       LDA #5E
00/030E: 85 3F       STA 3F
00/0310: A9 00       LDA #00
00/0312: 85 42       STA 42
00/0314: A9 20       LDA #20
00/0316: 85 43       STA 43
00/0318: 38          SEC
00/0319: 20 11 C3    JSR C311
00/031C: 60          RTS
00/031D: 00 00       BRK 00
00/031F: FF FF 00 00 SBC 0000FF,X
00/0323: FF FF 00 00 SBC 0000FF,X
00/0327: FF FF 00 00 SBC 0000FF,X
00/032B: FF FF 00 00 SBC 0000FF,X

*
```

Appendix

TABLE APP. 1. Page 1 and 2 Display Screens

Name	*Memory Addresses*		*Display Modes*
	Hex	*Decimal*	
1. Text/Low-Resolution Page 1 graphics	$400	1024	*Full Text Full Low-Resolution Mixed Text/ Low-Resolution
2. Text/Low-Resolution Page 2 graphics Program Memory*	$800	2048	Full Text Full Low-Resolution Mixed Text/ Low-Resolution

$ = Hexadecimal numbers
* = Most common usage
Note: Text/Low-Resolution graphics Page 2 can be used for screen display, but in practice it is rarely used for anything but Applesoft BASIC program storage.

TABLE APP. 2. High-Resolution Display Screens

Name	*Memory Addresses*		*Display Modes*
	Hex	*Decimal*	
3. High-Resolution Page 1 graphics	$2000	8192	Full High-Resolution *Mixed Text/ High-Resolution
4. High-Resolution Page 2 graphics	$4000	16384	*Full High-Resolution Mixed Text/High- Resolution graphics

$ = Hexadecimal numbers
* = Most common usage

TABLE APP. 3. Super High-Resolution Display Screens

Name	*Memory Addresses*		*Display Modes*
	Hex	*Decimal*	
8. Super High-Resolution 1	$E1/ 2000	225/ 8192	Full Super-Resolution *(320 Mode)
9. Super High-Resolution 2	$E1/ 2000	225/8192	Full Super-Resolution (640 Mode)

$ = Hexadecimal numbers
* = Most common usage

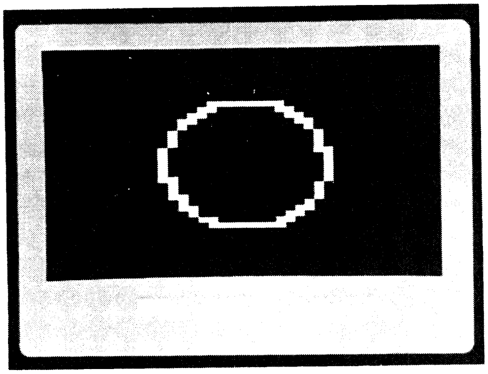


FIGURE App. 1. Graphic resolution comparison. Low resolution.

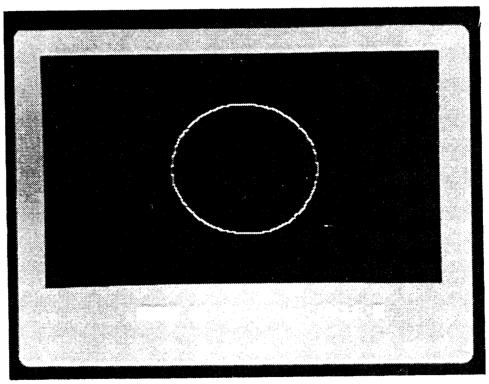


FIGURE App. 2. Graphic resolution comparison. High resolution.

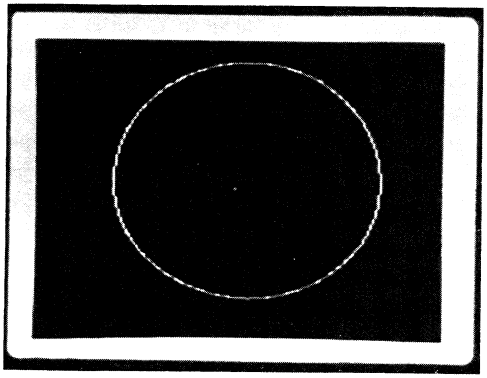


FIGURE App. 3. Graphic resolution comparison. 320-mode super high resolution.

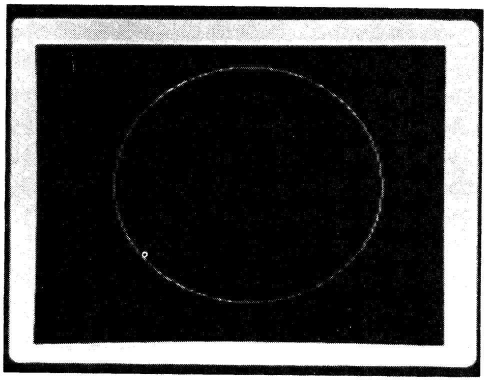


FIGURE App. 4. Graphic resolution comparison. 640-mode super high resolution.

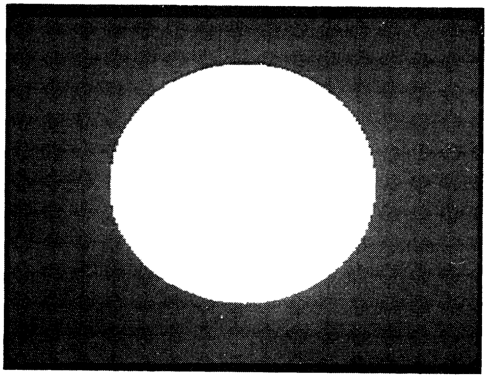


FIGURE App. 5. Graphic resolution comparison. Fill mode in super high resolution.

Answers to Questions

Chapter 2

1. Text and low-resolution graphics
2. True
3. False
4. PRINT
5. GR
6. COLOR
7. PLOT
8. TEXT

Chapter 3

1. HGR
2. HGR2
3. HCOLOR
4. HPLOT
5. FALSE
6. FALSE
7. Clear the text screen and place the cursor in the upper left corner of the screen.
8. 280 points (numbered 0 to 279)
9. 160 points (numbered 0 to 159)
10. 192 points (numbered 0 to 191)
11. POKE
12. Immediate mode
13. Deferred mode
14. RUN

Chapter 4

1. 8-bit and 16-bit or classic and native
2. 320 mode and 640 mode
3. 200 vertical lines in either mode
4. False, they both share the same memory location.
5. True

Chapter 5

1. 280 columns
2. 40 columns
3. 16 colors
4. 6 colors, but 8 numbers (2 black numbers and 2 white numbers)
5. High-resolution

Chapter 6

1. Draws a horizontal line from one column to another column on a specified row.
2. Draws a vertical line from one row to another row on a specified column.
3. Displays a point at the specified row and column
4. Determines the color of a specified screen position
5. True
6. 65 to 90
7. Peek (−16384). See the program listings
8. Poke −16368,0. See the program listings

Chapter 7

1. HPLOT or HPLOT TO
2. Column
3. 0,0
4. 279,191
5. False
6. False
7. Draw, erase, re-draw
8. POKE 230,32 = draw on Page 1
   POKE 230,64 = draw on Page 2
9. 6K or 6000 characters
10. BSAVE (Remember, the A, or A$ parameter specifies the starting point and the L or L$ parameter indicates the length.)

Chapter 8

1. DRAW and XDRAW
2. SCALE
3. ROT
4. Vector

5. A shape table
6. The number of shapes in the shape table
7. Zero
8. POKE 232,____ POKE 233,____
9. BSAVE filename, A$ starting address, L$ length of shape
10. BLOAD or BLOAD A$ location in memory to place shape table.

Chapter 9

1. A definition by screen coordinates and a definition by graph coordinates.
2. High-resolution character SHAPES.
3. True
4. Deviation graph
5. Diskettes or disk files

Chapter 10

1. El
2. POKE 49205,0
3. POKE 49193,163
4. Scanline control byte
5. False
6. 16 color tables are possible at any one time
7. Each color table can have up to 16 colors in it.

Chapter 11

1. Bit 7
2. Bit 4
3. False
4. False
5. False
6. The TOOLBOX or toolset

Index

>$18.95

MASTERING APPLESOFT GRAPHICS

"**MASTERING APPLESOFT GRAPHICS** offers probably the most detailed and comprehensive look at programming graphics for the Apple II series. Other books on the market do not compare to **MASTERING APPLESOFT GRAPHICS** with its in-depth approach utilizing real-life examples and its wealth of program listings." – Mark D. Veljkov, PhD — Computer Information Specialist, Bureau for Faculty Research, Western Washington University

Applesoft BASIC remains a constant across all the Apple *II* family computers, and **MASTERING APPLESOFT GRAPHICS** will help you learn to create graphics on your machine. And if you upgrade, your expertise will transfer to the new computer. The many examples, program listings, screen facsimiles, and graphic illustrations will accelerate your proficiency. This comprehensive, advanced treatment of Applesoft Graphics also is ideal for the experienced programmer who wants to learn more about the inner workings of the Apple IIGS.

Written in a reassuring, easy-to-read style, **MASTERING APPLESOFT GRAPHICS** takes you from common mistakes to avoid to high-resolution screens to advanced Apple IIGS graphic information. Each chapter has questions and answers to help you chart your progress. Finally, here is a book to help you obtain the maximum from your computer's graphic capabilities.

David Miller is an experienced educator, software developer, computer book writer, and computer consultant. This is his fifth computer book, and its readability is the result of his classroom experience teaching at all levels, from introductory computer courses to programming courses. Mr. Miller consults frequently for businesses on programming, setting up computer systems, and training users.

ISBN 0-673-38148-X

9 780673 381484

Scott, Foresman and Company